Universitat de Lleida

# TREBALL FINAL DE GRAU

ESCOLA POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: Marc Viladegut Abert

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: **Manipulació d'objectes amb interfícies naturals d'usuari**

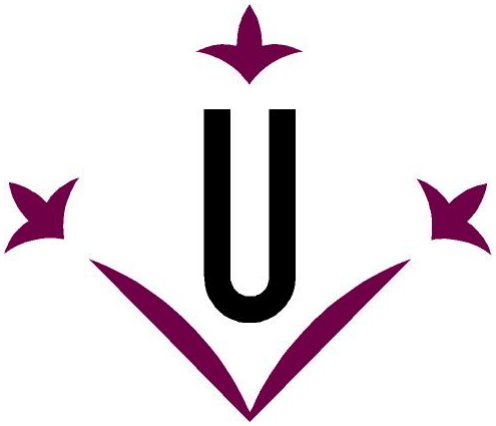
Director/a: **Toni Granollers Saltiveri**
**Juan Enrique Garrido Navarro**

Presentació

Mes: Juliol

Any: 2019

UNIVERSITAT DE LLEIDA



TREBALL DE FINAL DE GRAU

# MANIPULACIÓ D'OBJECTES AMB INTERFÍCIES NATURALS D'USUARI

*Autor:*
MARC VILADEGUT ABERT

*Tutor/s:*
DR. TONI GRANOLLERS
DR. JUAN ENRIQUE GARRIDO

*Un treball de final de grau presentat per superar el grau en Enginyeria informàtica*

*en el*

DEPARTAMENT D'INFORMÀTICA
ESCOLA POLITÈCNICA SUPERIOR

9 DE JULIOL DE 2019

UNIVERSITAT DE LLEIDA

# *Introducció*

Escola Politècnica Superior
Departament d'informàtica

**Manipulació d'objectes amb interfícies naturals d'usuari**

per Marc VILADEGUT ABERT

Avui en dia, en un món que no para d'evolucionar i les tecnologies avancen a un ritme que és quasi impossible de seguir, la interacció entre persones i màquines ha estat present al llarg del temps des de la creació del primer dispositiu, fins avui en dia. Naturalment, per comunicar-se amb les màquines, sempre ha estat necessari algun dispositiu d'entrada que fos capaç de provocar-li un *feedback* com pot ser un teclat o un ratolí.

En aquests últims anys s'ha investigat les maneres d'interaccionar amb un sistema sense cap mitjà que impliqui comunicar-se utilitzant un dispositiu d'entrada. Aquest tipus d'interacció s'anomena interacció natural, on la comunicació persona i màquina es pot realitzar per mitjà dels moviments, veu o altres maneres d'involucrar els sentits de l'ésser humà sense utilitzar el tacte (pantalles tàctils). Al llarg del temps aquest tipus d'interacció ha tingut una gran repercussió i més en el món dels videojocs. Grans indústries, avui en dia, segueixen apostant per aquest tipus d'interacció per treure-li tot el potencial que sigui possible, ja no únicament en videojocs, que és quan més impacte va causar arreu del món, sinó que també en nous camps de la investigació en què podria ser de gran utilitat.

Gràcies a l'aparició d'aquests dispositius que fan possible la interacció de forma natural, s'ha obert un món de possibilitats en l'aparició de videojocs, sobretot aquells que són controlats en primera persona i que utilitzen algunes tecnologies com són el núvol de punts per imitar els moviments de l'usuari dins del videojoc. Als usuaris se'ls permet interactuar amb l'entorn virtual i els diversos elements que el formen, aportant així una major experiència d'usuari i més realisme a l'hora d'interactuar. Aquests tipus de videojocs es poden veure també avui en dia, en les tecnologies immersives com pot ser la realitat virtual, això sí, amb la utilització de dispositius d'entrada.

## Motivació

Es podria dir que la causa principal per dur a terme aquest projecte va ser el repte a enfrontar-se a les noves tecnologies, és a dir, aquelles tecnologies que s'han anat descobrint al llarg d'aquests últims anys i sabent que algunes d'elles han causat gran impacte en la societat d'avui en dia.

Tanmateix, una reunió amb els professors Juan Enrique Garrido i en Toni Granollers va ser suficient per decantar-se per aquest tipus de tecnologia com és la interacció natural. Inicialment no es tenia cap idea de projecte en ment, únicament se sabia que es volia treballar amb aquest tipus de tecnologia. El Juan Enrique Garrido ja havia treballat en aquest camp tecnològic diverses vegades[1][2], així que ja era coneixedor d'aquesta interacció. Va ser crucial el seu coneixement, ja que va poder oferir idees de projectes molt diverses, les quals una d'elles s'ha posat en pràctica en aquest projecte.

S'ha estat coneixedor des del primer moment de la dificultat que podia suposar aquest projecte, i tota la càrrega de treball que implicava. Encara així s'ha seguit apostant per realitzar-lo. La possibilitat d'incloure investigació i de poder descobrir les formes d'interacció a partir d'una càmera 3D, han estat els grans motius de l'elecció del projecte.

## Objectius

L'objectiu principal del projecte, exigeix presentar la fase inicial de desenvolupament d'un sistema, on l'objectiu principal és treballar la manipulació d'objectes 3D en un entorn virtual mitjançant la interacció basada en moviment amb un dispositiu de nova generació. En concret, es pretén modificar la posició d'un objecte situat en un escenari tridimensional (representat en una pantalla 2D), movent-lo, arrossegant-lo i fins i tot, modelar la seva forma, tot això de manera no invasiva, és a dir, sense la necessitat de què l'usuari transporti en el seu cos algun dispositiu.

Per fer-ho possible serà necessari incorporar en el sistema una càmera 3D que ofereixi aquest tipus d'interacció natural i que s'adapti als requeriments que suposa aquest projecte. Serà necessari investigar l'actualitat del mercat pels dispositius que interaccionen amb NUI, saber quins dispositius existeixen i si encara se segueix apostant per ells. Caldrà esbrinar tant els avantatges com els inconvenients del SDK que incorpora i si, en un futur, seria necessari optar per altres alternatives de tercers per substituir o millorar els serveis ofert. Posteriorment, caldrà esbrinar com s'utilitzarà per realitzar el projecte, i també el software requerit pel desenvolupament i els llenguatges de programació que implica l'ús d'aquests.

El fet de desenvolupar un projecte d'aquestes característiques serà beneficiós tant en termes d'adquirir coneixements sobre noves formes d'interacció amb els dispositius, com també llenguatges de programació. Serà imprescindible el coneixement de nous IDE que no s'han vist fins al moment per dur-ho a terme.

## Descripció del projecte

Aquest projecte té com a finalitat, l'entrega de la fase inicial de desenvolupament d'un sistema on es treballaran els objectius esmentats en la secció anterior. Així mateix per arribar al final d'aquest projecte serà necessari la realització dels diversos objectius en qüestió.

Inicialment i respecte al sensor que permetrà la interacció basada en moviment, es realitzarà prèviament una cerca exhaustiva per veure quin és el millor dispositiu que es troba actualment en el mercat i que encaixa amb els requisits del projecte. Respecte al SDK del dispositiu, serà necessari una investigació per esbrinar quin és el que s'adapta millor als objectius del projecte i si fos necessari, incloure altres softwares de tercers per augmentar l'enriquiment del sistema. El fet de buscar un SDK que compleixi els objectius del projecte implicarà, cercar un IDE que també ho faci, ja que serà on es durà a terme tot el desenvolupament del projecte i serà l'encarregat de rebre, analitzar i actuar en conseqüència de tot el *feedback* que provingui del dispositiu.

El prototip que es desenvoluparà serà d'escriptori i haurà de detectar la càmera que estarà connectada via USB. Una vegada s'inicialitzi, caldrà detectar el jugador que es trobarà dins del rang d'actuació del dispositiu i posteriorment captarà els possibles moviments que pugui realitzar. Cada moviment tindrà un *feedback* en l'aplicació, ja sigui únicament de moure el cursor d'interacció, manipular la figura, etc.

El desenvolupament del prototip es dividirà en dos mòduls que tindran connexió entre ells. El primer d'ells que fa referència a la part de l'Òscar López, s'encarregarà de les interfícies d'usuari que tingui el prototip, i també realitzar el disseny de totes elles centrat en l'usuari. En canvi, l'altre mòdul i que s'explicarà en aquesta memòria, s'encarregarà de realitzar la manipulació de la figura 3D en l'escena tridimensional. La idea és que al final del desenvolupament, s'integraran els dos mòduls en un únic prototip on posteriorment es duran a terme diverses avaluacions amb usuaris per obtenir les conclusions finals sobre el projecte.

L'ús d'aquest prototip està orientat per aquelles persones que volen entrenar algunes capacitats específiques, millora d'algunes habilitats, etc. És possible utilitzar-la com a entreteniment per als més petits o també, en àmbits de recerca, inclús de rehabilitació, per als més grans. Cada funció deprendrà del context que se li vulgui donar al prototip.

## Estructura del document

L'estructura que fa referència a la memòria del projecte està formada pels següents capítols:

- En el **capítol 1** es parlarà de les interfícies d'usuari, la seva evolució, els diferents paradigmes d'interacció i els diferents tipus d'interfícies que existeixen avui en dia, ja que és un dels temes principals dels quals tracta aquest projecte. També es farà èmfasi sobre les interaccions basades en moviment, els dispositius que inclouen aquesta interacció i les comparatives sobre els diferents dispositius *hardware* que existeixen al mercat d'avui en dia.

- En el **capítol 2** s'explicarà la informació del sensor i la utilització d'un *middleware* per la comunicació entre el sistema operatiu i l'IDE en el qual es desenvoluparà el projecte. Aquest software serà un motor de joc en el qual es durà a terme, durant tot el projecte, tot el desenvolupament del prototip. Per finalitzar, s'explicarà la forma de comunicar-se amb el sensor, les seves característiques i totes aquelles eines addicionals que s'utilitzaran i que formaran part del desenvolupament del projecte.

- En el **capítol 3** s'explicarà la metodologia que es durà a terme en el desenvolupament d'aquest projecte, com s'utilitzarà i també s'explicaran algunes metodologies Agile que s'afegiran abans de l'inici del desenvolupament. Tot seguit, es farà èmfasi de les decisions de disseny i d'implementació abans d'iniciar el desenvolupament basat en iteracions.

- En el **capítol 4** tracta de tot el desenvolupament basat en iteracions que farà referència a la part de la manipulació dels objectes 3D en l'entorn tridimensional. S'explicaran tots i cadascun dels passos que es vagin duent a terme en el projecte, siguin correctes o erronis i el perquè de cada decisió. S'efectuaran totes i cadascuna de les iteracions que estiguin previstes i si és necessari, s'afegiran de noves. De cada iteració s'explicarà l'anàlisi, el disseny, el desenvolupament i les dificultats trobades durant tot el procés de desenvolupament.

- Finalment, en el **capítol 5** s'expressarà les conclusions que s'han tret després de dur a terme aquest projecte i les possibles línies de futur sobre el prototip final, siguin millores o accions futures.

# Planificació i viabilitat del projecte

## Planificació temporal

Respecte a la planificació del projecte, s'han agrupat els diversos objectius del projecte en les iteracions que apareixeran en la següent taula 1, on es pot observar tota la vida del projecte reduïda amb 7 tasques principals on el desenvolupament d'algunes s'ha dut a terme de forma paral·lela amb l'altre mòdul. La durada de cada tasca és una aproximació del temps total, ja que ha resultat impossible saber amb certesa el temps exacte en cada tasca.

| Id. | Nom de la tasca | Inici | Fi | Durada |
|---|---|---|---|---|
| **1** | Configuració de Orbbec Astra Pro | 04/12/2018 | 17/12/2018 | 14 dies |
| **2** | Configuració de Nuitrack | 18/12/2018 | 01/01/2019 | 14 dies |
| **3** | Iteració 1 | 02/01/2019 | 22/02/2019 | 52 dies |
| **4** | Iteració 2 | 23/02/2019 | 29/03/2019 | 35 dies |
| **5** | Iteració 3 | 30/03/2019 | 29/05/2019 | 61 dies |
| **6** | Realització dels informes | 01/02/2019 | 22/05/2019 | 112 dies |
| **7** | Realització de memòria | 07/12/2018 | 14/06/2019 | 184 dies |

TAULA 1: Planificació temporal de les *issues*

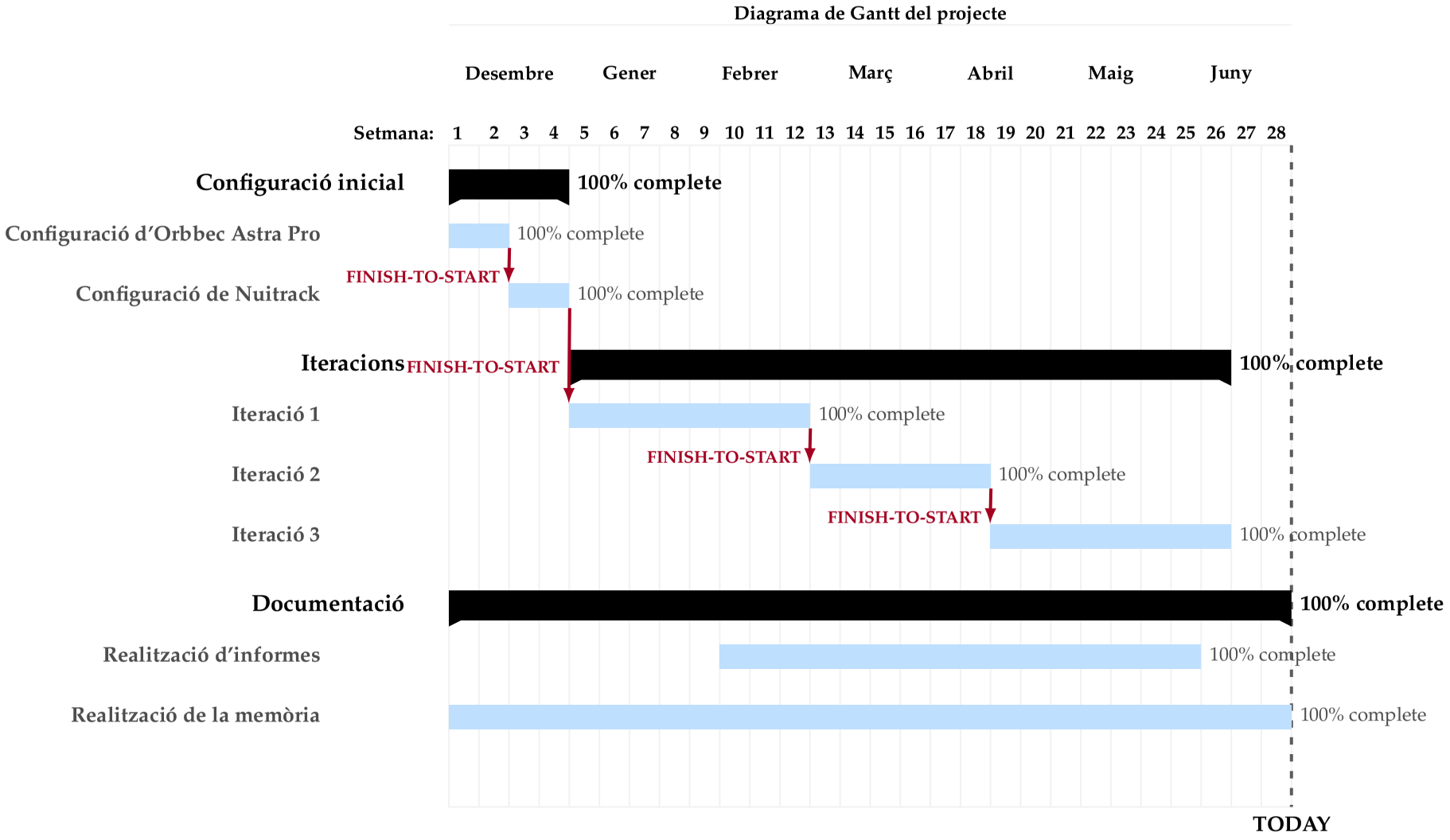



FIGURA 1: Diagrama de Gantt del treball de final de grau

Posteriorment, s'ha generat el diagrama de Gantt que fa referència a les durades introduïdes per cadascuna de les tasques que apareixen en la taula 1. Com es pot observar en el diagrama de Gantt de la figura 1 anterior, les tasques inicials es duran a terme de forma iterativa (sense tenir en compte la realització de la memòria).

En referència a la totalitat del projecte, el mòdul que es durà a terme en aquesta memòria, no s'encarregarà ni la integració de les parts, ni l'avaluació final del prototip. Únicament es treballarà en el desenvolupament i la recerca dels diferents objectius del mòdul de la *manipulació de l'objecte 3D en un entorn tridimensional*. L'altre mòdul es desenvoluparà de forma independent al projecte en qüestió i quan s'acabi el desenvolupament d'aquest serà l'altre mòdul el que s'encarregarà tant de la integració, com de la corresponent avaluació final.

## Estudi de viabilitat

En l'estudi de viabilitat s'ha de tenir present que el prototip del projecte, ha de ser un programa senzill i intel·ligible, que qualsevol usuari en pugui fer ús i entendre'l sense cap mena de dificultat.

En la taula 2 següent, es podrà observar l'anàlisi DAFO[1] que s'ha dut a terme abans d'iniciar el projecte. Analitzant aquesta taula es pot aprofitar aquestes oportunitats i fortaleses per desenvolupar el prototip, encara que hi hagi alguna debilitat i amenaça que pugui fer que el prototip no acabi de funcionar del tot bé o de ser de gran utilitat.

Els riscos que poden aparèixer durant el desenvolupament, l'intent d'arreglar o tapar aquests riscos pot causar la pèrdua de moltes hores de treball. La possibilitat de no saber com resoldre determinades dificultats o l'aparició d'errors sense saber-ne la causa exacta pot suposar un risc important en el projecte, ja que això implicarà en un futur, un canvi d'iteració en el desenvolupament.

Un dels riscos a tenir més en compte, és el temps. Es tenia coneixement, que quan es va escollir aquest projecte, es treballaria en coses innovadores, les quals no s'han dut a terme anteriorment, fet que implica, dedicar moltes hores en la investigació i menys en el desenvolupament. Existeix la possibilitat que no es completin els objectius proposats inicialment per dur a terme en la fase de desenvolupament.

Per acabar, el cost d'aquest projecte ve lligat per la inversió en l'equipament de treball i les hores dedicades en el desenvolupament d'aquest. Cal tenir en compte que el projecte global és dur a terme en 2 mòduls on cadascun d'ells estarà format per 1 persona, per tant el preu pot pujar més de l'habitual. Posant preu a les hores invertides, a la realització d'aquesta memòria i a tot l'equipament, el pressupost d'aquest mòdul del projecte ascendeix aproximadament a 3.100€.

[1] Anàlisis DAFO. (2018, 12 de diciembre). Wikipedia, La enciclopedia libre. Fecha de consulta: 14:07, diciembre 26, 2018 desde https://es.wikipedia.org/w/index.php?title=An%C3%A1lisis_DAFO&oldid=112619939.

| | Positiu (per assolir l'objectiu) | Negatiu (per assolir l'objectiu) |
|---|---|---|
| Origen intern (producte/atributs de l'empresa) | ☛ Desenvolupament parcial de l'aplicació amb grup. Més difícil d'elegir decisions errònies.<br>☛ Gran actitud d'equip i treball en equip. Amb capacitats de sacrifici i creativitat.<br>☛ Gran capacitat resolutiva per part dels membres del grup. | ☛ Desconeixement sobre el desenvolupament d'aplicacions amb càmeres de la sèrie Orbbec Astra.<br>☛ Desconeixement sobre l'utilització dels diferents IDEs per desenvolupar el videojoc.<br>☛ El SDK incorporat no disposi de totes les tecnologies necessàries per dur a terme la manipulació del objecte 3D. |
| Origen extern (entorn/atributs del mercat) | ☛ Desenvolupament del videojoc en el SO de Windows on disposa de gran quantitat d'usuaris.<br>☛ Manca d'existència d'aplicacions en Windows que puguin generar competència i reduir les vendes/baixades.<br>☛ Nou desenvolupament amb sensors 3D que apenes s'havia treballat fins al moment. | ☛ El videojoc requereix d'un sensor 3D per interactuar-hi.<br>☛ Desconeixement en part d'aquest tipus de tecnologia per part de molts usuaris.<br>☛ Usuaris molt exigents i el conjunt d'usuaris que l'utilitzen és molt limitat. |

TAULA 2: Diagrama DAFO del projecte

## Agraïments

Aquest projecte està recolzat per la beca d'investigació de la Universitat de Lleida "Convocatòria de beques d'introducció a la investigació de la Universitat de Lleida per al curs 2018-2019 (acord núm. 165/2018 del Consell de Govern del 19 de juny)". La realització ha estat al laboratori del qual disposa el grup GRIHO a l'Escola Politècnica Superior. Agrair també a tot l'entorn, ja sigui família, amics, no tan amics i professors pel suport moral que m'han donat aquests últims mesos de treball i per ensenyar-me a no deixar de lluitar en els moments més importants.

# Índex

# Índex de figures

# Índex de taules

# Llista d'abreviatures

| | |
|---|---|
| **API** | **A**pplication **P**rogramming **I**nterface |
| **BCI** | **B**rain **C**omputer **I**nterface |
| **CLI** | **C**ommand **L**ine **I**nterface |
| **DAFO** | **D**ebilitats **A**menaces **F**ortaleses **O**portunitats |
| **DLL** | **D**ynamic-**L**ink **L**ibrary |
| **GIF** | **G**raphics **I**nterchange **F**ormat |
| **GUI** | **G**raphical **U**ser **I**nterface |
| **IDE** | **I**ntegrated **D**evelopment **E**nvironment |
| **NUI** | **N**atural **U**ser **I**nterface |
| **PUI** | **P**erceptual **U**ser **I**nterface |
| **SDK** | **S**oftware **D**evelopment **K**it |
| **SO** | **S**istema **O**peratiu |
| **TUI** | **T**angible **U**ser **I**nterface |
| **TUI** | **T**ext **U**ser **I**nterface |
| **UI** | **U**ser **I**nterface |
| **UX** | **U**ser e**X**perience |
| **VUI** | **V**oice **U**ser **I**nterface |
| **XP** | e**X**treme **P**rogramming |

*Even the best designers produce successful products only if their designs solve the right problems. A wonderful interface to the wrong features will fail.*

*Jakob Nielsen*

# 1 Les interfícies i les seves interaccions

## 1.1 | Interfície d'usuari

La comunicació entre persona i ordinador ha estat present des de la creació del primer sistema fins avui en dia gràcies a l'aparició de les interfícies. Tanmateix, al llarg dels últims anys ha existit una evolució molt notòria entorn aquestes.

### 1.1.1 | Què és?

S'anomena interfície d'usuari o ***U**ser **I**nterface* (UI) al mitjà a través del qual la persona humana es pot comunicar amb la màquina i controlar un software o hardware específic. El que s'intenta aconseguir normalment en una interfície d'usuari és que sigui fàcil d'utilitzar, ja que així és més instintiva i intuïtiva. Avui en dia, es pot trobar interfícies d'usuari en quasi tots els llocs on existeix tecnologia digital, des de telèfons mòbils, automòbils, ordinadors, reproductors de música, smart TV, etc.

Els objectius principals que tenen les interfícies d'usuari bàsicament són dos, el primer d'ells és que una interfície d'usuari ha de poder comunicar informació a l'usuari mitjançant algun tipus de dispositiu o sistema. En el segon, una vegada s'ha completat el primer, es vol aconseguir que la comunicació sigui la més còmoda i fàcil possible per l'usuari que utilitza l'aplicació.

Tanmateix, les interfícies d'usuari es poden classificar de 3 tipus que són:

- **Interfície de hardware**. La comunicació es durà a terme mitjançant perifèrics per introduir, processar i entregar les dades. Els exemples més bàsics d'aquest tipus d'interfície podria ser el teclat, ratolí i el monitor.
- **Interfície de software**. És la interfície d'una aplicació que resideix en una màquina on la finalitat de l'aplicació, és comunicar-se amb la màquina per a poder extreure resultats/informació.

- **Interfície de software-hardware**. És una interfície on s'estableix una connexió entre la màquina i les persones. Això permet a les dues parts, entendre entre ells les dificultats que hi poden haver entre els dos idiomes (màquina i ésser humà).

### 1.1.2 | Evolució

Al llarg de la història de la informàtica, les interfícies d'usuari han adoptat canvis molt notoris des dels inicis de la informàtica fins avui en dia. Aquests canvis es veuen reflectits en la manera la qual l'ésser humà ha anat interaccionant amb la màquina al llarg del temps (vegeu figura 1.1).


FIGURA 1.1: L'evolució de les interfícies d'usuari
Font: https://es.wikipedia.org

Tot seguit, es veurà de forma més detallada les interfícies que componen aquesta evolució al llarg del temps i que apareixen en la figura 1.1 anterior:

– ***Command-Line Interface* (CLI)**. És la interfície que existeix gairebé des dels principis de la informàtica, és la més bàsica de totes. Està composta únicament de text pla i la comunicació amb la màquina en aquesta interfície es dóna mitjançant comandes. És una manera de comunicar-se potent i flexible, però és una interacció difícil d'aprendre donat que s'han de memoritzar les comandes per interactuar. Aquest tipus d'interfície ja s'ha vist reemplaçada per les interfícies gràfiques i avui en dia, aquest tipus d'interacció és suplementaria.

– ***Graphical User Interface* (GUI)**. És l'evolució de la interfície de línia de comandes. Aquesta interfície disposa d'una part gràfica on l'usuari interactua amb la màquina mitjançant diversos objectes gràfics els quals proporciona a l'usuari accions i informació (*feedback*). La introducció d'aquesta interfície va aportar més facilitats per interactuar amb les màquines, proporcionant així un entorn visual més senzill respecte la CLI i més fàcil d'aprendre per part de l'usuari. És la més utilitzada al llarg dels anys, tant és així, que s'han introduït pautes que ajuden a millorar la facilitat d'ús d'una interfície GUI, un clar exemple és les 8 regles d'or de Ben Shneiderman[1].

[1] *8 Golden Rules for Better Interface Design*. (2018, 26 de abril). *Envato tuts, Free How-To Tutorials & Online Courses by Envato Tuts+*. Data de consulta: 18:15, desembre 25, 2018 des de https://webdesign.tutsplus.com/articles/8-golden-rules-for-better-interface-design--cms-30886

– ***Natural User Interface*** **(NUI)**. És el tipus d'interfície més nova que existeix avui en dia, no es disposa de cap dispositiu d'entrada en el sistema, únicament s'utilitza, els sentits de l'ésser humà per comunicar-se (no tots, de moment) com podria ser el moviment del cos humà, la veu, els ulls, etc. Això és un gran avantatge, ja que hi ha un coneixement previ del mateix llenguatge i per tant és possible conèixer la forma d'interacció molt fàcilment.

Existeixen diverses variants dels tipus d'interfícies que han aparegut al llarg de la història. Aquestes principalment són la *Voice User Interface* (VUI), *Tangible User Interfaces* (TUI), *Text User Interface* (TUI), *Perceptual User Interface* (PUI) i *Brain-Computer Interface* (BCI), entre d'altres. Moltes d'elles actualment estan en fase d'investigació, ja que poden ser molt resolutives en molts camps de la investigació[3].

### 1.1.3 | Paradigmes d'interacció

Actualment els paradigmes d'interacció representen els exemples o models dels quals deriven tots els sistemes d'interacció explicats en la secció 1.1.2. Existeixen 4 paradigmes actualment i són els següents:

- **Ordinador de sobretaula**. És el paradigma dominant actualment. La interacció es realitza aïllat de l'entorn i assegut en una taula amb l'ordinador on les interfícies són de manipulació directa.
- **Realitat virtual** (VR). És el paradigma que està tenint bastant repercussió sobretot en el món dels videojocs avui en dia. La idea és que l'usuari interactua amb el sistema on existeix una sensació de presència dins de l'entorn virtual i aquesta és molt similar o igual al món real. El principal desavantatge d'aquesta tecnologia és pels problemes ocasionats en l'usuari que l'utilitza, arribant a provocar cansament o marejos en alguns casos.
- **Computació ubiqua**. És aquell paradigma que tracta d'estendre la capacitat computacional al voltant de l'usuari on la seva interacció és simple o inclús invisible.
- **Realitat augmentada**. En aquest paradigma l'usuari serà capaç d'interaccionar amb el món real on semblarà augmentat per la informació sintètica de l'ordenador. És dels paradigmes menys desenvolupat de tots, ja que requereix una demanda tecnològica molt gran i que encara no s'ha desenvolupat en gran mesura.

Avui en dia, tant l'ordinador de sobretaula com la realitat virtual són els dos paradigmes que estan més desenvolupats que la resta. En canvi, la computació ubiqua i la realitat augmentada són paradigmes que han anat sortint aquests últims anys i estan en ple desenvolupament avui en dia. Sobretot la primera d'elles on en temes de videojocs, l'empresa Apple Inc. va mostrar en la presentació WorldWide Developers Conference (WWDC), un joc juntament amb l'empresa LEGO. En aquesta conferència es podia observar que es jugava en realitat augmentada a sobre d'una taula on hi havia una construcció de LEGO i des d'un telèfon mòbil es podia emular una ciutat virtual[4][5].

### 1.1.4 | Natural User Interface (NUI)

Les interfícies naturals d'usuari (NUI) són de les últimes interfícies en aparèixer al mercat, ja que van néixer a la dècada de 1990 a les mans de Steve Mann on va crear una sèrie d'estratègies per la interfície d'usuari utilitzant la interacció natural amb el món real com una alternativa a les CLI i GUI que estaven presents en aquells temps[6].

FIGURA 1.2: Interacció amb una interfície natural d'usuari
Font: http://www.technokrata.hu/

Dels exemples més reconeguts en aquest tipus d'interfície, molts d'ells es troben dins del tipus VUI, on es produeix la comunicació mitjançant la veu per part de l'usuari i la màquina. Els aparells més coneguts són Siri[2] de l'empresa Apple Inc. o també Google Now[3] de l'empresa Google LLC.

Dels aparells en què es farà més èmfasi, ja que hi deriva una relació directa amb el projecte, és Kinect[4] de Microsoft Corporation. Aquesta càmera va sortir al mercat com una gran novetat, ja que permetia jugar a videojocs sense la necessitat de cap dispositiu d'entrada per interactuar. Per utilitzar aquest tipus d'interacció era necessari que la càmera pogués detectar l'ésser humà dins del camp d'acció i posteriorment, captar tots els moviments que realitzava. Els sistemes que treballen amb aquest tipus d'interacció, tracten de captar els moviments que realitza l'usuari mitjançant dispositius especialitzats, provocant així un *feedback* en el sistema per cada moviment que l'usuari realitzi (vegeu figura 1.2).

[2]Siri. (2018, 26 de desembre). *Wikipedia, La enciclopedia libre*. Data de consulta: 11:19, desembre 26, 2018 des de https://es.wikipedia.org/w/index.php?title=Siri&oldid=112876359.

[3]Google Now. (2018, 13 de desembre). *Wikipedia, La enciclopedia libre*. Data de consulta: 10:56, gener 14, 2019 des de https://es.wikipedia.org/w/index.php?title=Google_Now&oldid=112639466.

[4]Kinect. (2018, 14 de novembre). *Wikipedia, La enciclopedia libre*. Data de consulta: 11:13, desembre 26, 2018 des de https://es.wikipedia.org/w/index.php?title=Kinect&oldid=111992157.

En el cas de la figura 1.2, la càmera capta els moviments i els transforma en respostes on provoca moviments dins de l'entorn virtual. Com s'observa, no existeix cap dispositiu d'entrada que realitzi la comunicació, sinó que és la mateixa càmera que detecta l'ésser humà com un interactuador directe amb el sistema. Aquest tipus d'interacció es coneix avui en dia com interacció basada en moviments.

Avui en dia, la NUI és un tipus d'interacció que es troba en molts camps de la investigació, ja que existeixen moltes formes d'interaccionar i actualment, moltes d'elles, encara busquen ser útils per l'usuari en el context en què es trobi. Aquestes interfícies encara requereixen més investigació per millorar i consolidar-se en un mercat de manera estable, com podria ser en l'àmbit dels videojocs, o inclús, en l'àmbit de la rehabilitació, ja que és possible arribar a supervisar exercicis de rehabilitació per mitjà d'un sistema sense la necessitat d'un professional[1].

## 1.2 | Interacció basada en moviment

Dins de l'àmbit de la interacció natural, la interacció basada en moviment seria una forma d'interactuar de les moltes possibles que es trobarien dins de les interfícies naturals d'usuari i que és una altra forma d'interacció com ho són, les esmentades anteriorment, VUI, TUI, PUI, etc. Té com a avantatge el fet que l'usuari, en general, fa ús de moviments quotidians per interaccionar. És a dir, que és alguna cosa que es coneix, que no ha d'aprendre i que, per tant, ajuda a eliminar càrrega mental.

### 1.2.1 | Què és?

La interacció basada en moviment és una tècnica de comunicació amb el sistema a partir de la detecció i reconeixement dels moviments com s'ha explicat en la secció 1.1.4 en l'apartat de Kinect. Els moviments captats pel dispositiu seran suficients per provocar el *feedback* necessari per dur a terme una comunicació entre el sistema i l'usuari.

FIGURA 1.3: Sistema d'interacció basat en mètodes invasius
Font: https://en.wikipedia.org

Aquesta interacció, que està basada en la interpretació de moviments, permet a l'usuari controlar el sistema mitjançant els seus moviments amb qualsevol part del seu cos, sigui mitjançant gestos o moviments, en qualsevol cas naturals per un ample rang de persones. A més, és possible analitzar aquests moviments per, posteriorment, actuar en conseqüència. Detectar accions de l'usuari o postures, són alguns dels avantatges que ens ofereix aquest tipus d'interacció[2].

Existeixen diversos mètodes per dur a terme aquest tipus d'interacció com són els invasius i els no invasius. Per una banda, els invasius seria el fet de dur un dispositiu d'entrada enganxat a l'usuari com es pot observar en la figura 1.3, mentre que els no invasius és justament tot el contrari,

l'usuari no disposa de cap dispositiu d'entrada que estigui en contacte amb ell per provocar *feedback* (vegeu figura 1.2).

Per mitjà de l'ús de tècniques de computació, el sistema també permet interaccions respecte a la posició i rotació de l'usuari més pròxim al sensor. La llibertat de moviments que proporcionen aquests sistemes, sobretot en els no invasius, fa que l'usuari pugui navegar per l'escenari d'acció lliurement i sense dispositius que limiten el moviment[7].

Una variant d'aquest sistema pot ser, que no sigui l'usuari qui realitzi l'acció per comunicar-se amb el sistema, sinó que sigui el mateix sistema el que indiqui la posició que sigui necessari a adoptar per l'usuari, com podria ser el cas de la rehabilitació esmentat anteriorment, on és el mateix sistema el que porta el control de si l'usuari realitza bé els moviments[1].

### 1.2.2 | Dispositius d'interacció basats en moviment

En aquest camp d'interacció tan gran com és NUI, les eines que poden proporcionar aquesta interacció poden ser bastant diverses. Un dels elements principals per aquest tipus d'interacció (basada en moviment) pot ser una càmera, ja que s'encarregarà de detectar els moviments que realitzarà l'usuari, també existeixen altres dispositius que permeten detectar el moviment de, per exemple, les mans com és el cas del Leap Motion[5]. Tanmateix, el tipus de càmera que es pot adquirir pot oscil·lar des d'una càmera simple (similar a una càmera web) fins a un monitor amb una càmera incorporada amb la finalitat d'utilitzar-la en alguns alguns casos pel *face tracking*[6] o el *eye tracking*[7].

També existeix alguna variant com són sensors que s'introdueixen en les diferents articulacions de l'ésser humà per suplir la càmera 3D. Encara que sigui una tecnologia invasiva per l'ús de sensors, en diversos contexts pot ser una tecnologia molt precisa i és una bona variant per prescindir l'ús de càmeres.

Aquest tipus d'eines solen venir acompanyades amb el controlador proporcionat pel fabricant. També s'acostuma a proporcionar el SDK, que conté les llibreries necessàries per a poder utilitzar totes les tècniques possibles que ofereix l'eina i aquest tipus d'interacció. Normalment, cada fabricant té a disposició de l'usuari el seu propi SDK, no sempre gratuïts. Tanmateix, no es prescindirà, si és necessari, de l'ús de software de tercers que puguin aportar més enriquiment del sistema, ja sigui per introduir funcionalitats noves, com per aportar més avantatges al projecte.

[5]Leap Motion. (2018, 3 de desembre). *Wikipedia, The Free Encyclopedia*. Data de consulta: 19:19, Juliol 8, 2019, de https://en.wikipedia.org/wiki/Leap_Motion

[6]Facial recognition system. (2019, 15 de maig). *Wikipedia, The Free Encyclopedia*. Data de consulta: 10:08, Maig 30, 2019, de https://en.wikipedia.org/w/index.php?title=Facial_recognition_system&oldid=897267651

[7]Seguiment d'ulls. (2018, 17 d'agost). *Viquipèdia, l'Enciclopèdia Lliure*. Data de consulta: 09:30, desembre 29, 2018 de https://ca.wikipedia.org/w/index.php?title=Seguiment_d%27ulls&oldid=20210437.

### 1.2.3 | Cerca de dispositius

Per realitzar el projecte és necessari, com a mínim, una eina capaç de proporcionar interacció basada en moviment. Tot seguit, s'exploraran les diferents alternatives que existeixen avui en dia, per tal d'implementar aquest tipus d'interacció, sigui mitjançant una càmera o un sensor. També s'intentarà no tenir en compte sensors 3D com Kinect, ja que avui en dia ha quedat pràcticament obsolet[8] i s'apostarà per dispositius de nova generació que puguin oferir les mateixes o millors prestacions. Ha quedat demostrat també, que Kinect és perfectament reemplaçable per les càmeres de nova generació que es troben avui en dia al mercat [9].

Es durà a terme una cerca exhaustiva en els diversos fabricants més coneguts, especialitzats en dispositius d'aquest tipus d'interacció, on es tindrà en compte les diverses opinions de diferents usuaris que hagin apostat pel canvi de Kinect i les diferents pàgines de tecnologia que proporcionin informació de rellevància sobre nous dispositius que apareguin en el mercat. Es prioritzarà en tot moment de no escollir dispositius que siguin invasius, ja que la idea és reemplaçar el dispositiu Kinect que ja no era invasiu de per si, per un dispositiu de nova generació, que també no ho sigui.

### 1.2.4 | Comparació dels dispositius del mercat

Els models que s'han tingut en compte per l'elecció del dispositiu han estat únicament càmeres 3D, ja que sobretot estan especialitzades en la interacció basada en moviment i no requereixen dispositius invasius a priori. Una vegada s'ha dut a terme la cerca, les càmeres que s'han trobat són els que fan referència a la taula 1.1. Generalment, s'han escollit diversos fabricants mirant que les característiques i prestacions siguin molt diverses. Tanmateix, el rang de preus entre totes les càmeres és similar, a excepció d'algunes elles.

Com s'observa en la majoria de càmeres, moltes d'elles comparteixen una resolució i un camp de visió bastant similars entre tots els models, no obstant no es prioritzen molt aquests aspectes, ja que no se li ha de donar un ús molt sofisticat i no és necessària la qualitat d'imatge si tampoc es mostrarà en l'escena. En el que la profunditat del sensor respecta, aquesta varia en moltes d'elles, si és té en consideració els objectius del projecte, la profunditat del sensor també és secundari, ja que la major part de la interacció es realitza de forma estàtica.

Una cosa curiosa i que està bé tenir en compte, és que la majoria de les càmeres admeten el SDK de OpenNI[8]. Això en part, pot suposar un avantatge al sistema, ja que proporciona portabilitat. La part negativa de tot això, és que OpenNI fa anys que es va quedar obsolet i no és un *framework* en el que es pugui deixar caure el pes del projecte[10].

No es requereix, per l'ús que se li donarà, d'optar per una càmera amb unes característiques específiques, ja que totes elles són de valors similars i no varien molt en les seves característiques.

[8]OpenNI. (2017, 16 de març). *Wikipedia, La enciclopedia libre*. Data de consulta: 13:26, gener 14, 2019 des de `https://es.wikipedia.org/w/index.php?title=OpenNI&oldid=97608810`.

| **Model** | **Resolució** (píxels) | **Visió** (hor.×ver.) | **Prof. del sensor** (m) | **SDK** | **Preu** |
|---|---|---|---|---|---|
| Intel Realsense 435D | 1920×1080 | 86º×57º | 0,2 - 10 | Intel Realsense SDK 2.0 | 154,35 € |
| Kinect v2 | 1920×1080 | 70º×60º | 4,5 | Kinect SDK 2.0 | 159,00 € |
| Orbbec Astra | 640×480 | 60º×49,5º | 0,6 - 8 | Astra SDK o OpenNI | 129,42 € |
| Orbbec Astra S | 640×480 | 60º×49,5º | 0,4 - 2 | Astra SDK o OpenNI | 129,42 € |
| Orbbec Astra Pro | 1280×720 | 60º×49,5º | 0,6 - 8 | Astra SDK o OpenNI | 129,42 € |
| Primesense Carmine 1.09 | 1280×980 | 54º×45º | 0,35 - 3 | OpenNI | 254,38 € |
| Vico VR | 1280×720 | 60º×49,5º | 0,5 - 4,5 | Vico SDK o OpenNI | 214,71 € |
| Xtion Pro Live | 1280×1024 | 58º×45º | 0,8 - 3,5 | OpenNI | 258,69 € |
| Zed | 4416×1242 | 90º×60º | 0,5 - 20 | ZED SDK 2.7 | 387,17 € |

TAULA 1.1: Característiques de les càmeres que es troben actualment en el mercat

*"Everyone we interact with becomes a part of us."*

*Jodi Aman*

# 2

# El sensor, la comunicació i el motor de física

## 2.1 | Per què Orbbec Astra Pro?

De les possibles càmeres a tenir en compte en la taula 1.1 de la secció 1.2.4 la que s'ha cregut que encaixava més amb els objectius preestablerts en el projecte, és el model Astra Pro de l'empresa Orbbec 3D Technology International Inc.

Les característiques que incorpora són la resolució de 1280×720, on no es buscava cap càmera que fos capaç de gravar a una resolució superior, ja que per al desenvolupament del projecte no es tindrà en compte la qualitat de la imatge, sinó el seguiment dels moviments. Per altra banda, el camp de visió ($60^o \times 49{,}5^o$) no és dels més amplis, però serà suficient per al tipus de projecte que es desenvoluparà, ja que no requerirà molt moviment i per tant el camp de visió no afectarà en gran mesura. Respecte a la profunditat del sensor (0,6 – 8 metres), és de les càmeres que suporta un interval d'actuació molt ample, no la millor, però si de les millors i això tampoc és molt important, ja que els motius són els mateixos que en el camp de visió explicat anteriorment. Per últim en termes econòmics, és la més econòmica juntament amb la resta de models de la sèrie Astra de Orbbec.

El que ha fet descartar algunes de les diverses càmeres de la taula 1.1 és la compatibilitat amb OpenNI en el SDK, ja que com s'ha esmentat en l'anterior secció 1.2.4, la major part de les càmeres suporten aquest tipus de llibreria i això és bo en termes de compatibilitat. Si es volgués continuar el projecte en alguna línia de futur, el projecte es podria desenvolupar per mitjà d'una càmera diferent sempre que ofereixi suport per OpenNI. L'empresa Orbbec també proporciona diversos SDK i controladors per a poder desenvolupar projectes en diversos sistemes operatius sigui Windows, Linux o Mac. Per últim, també proporciona compatibilitat amb el SDK de Gestoos[1] on és un software de tercers que aporta més funcionalitat al sensor, on també incorpora diverses tècniques per poder interaccionar de forma natural però, malauradament no s'ha pogut tenir accés a provar aquest el SDK, ja que no hi ha hagut resposta a la sol·licitud necessària per obtenir-lo.

[1] Reinventing Interaction - Gestoos. https://gestoos.com/.

## 2.2 | Comunicació amb el sensor: Nuitrack™SDK

Com s'ha esmentat en la secció 2.1, la utilització OpenNI millora la portabilitat del sistema que es crearà, ja que com s'havia investigat anteriorment, la major part de les càmeres suportaven aquest tipus de *framework*. Encara que la part negativa d'aquest, és que actualment està obsolet, s'han buscat altres alternatives que poden intentar proporcionar compatibilitat al sistema amb diferents càmeres 3D.

No obstant això, s'ha investigat noves formes per dur a terme el projecte intentant mantenir la portabilitat que oferia OpenNI en el seu moment. S'ha trobat una variant que la manté i realitza una funció similar. Avui en dia, encara s'hi dóna suport i a poc a poc s'afegeix nou hardware el qual proporciona més compatibilitat. Aquest *middleware* s'anomena Nuitrack.

### 2.2.1 | Què és?

Nuitrack és un SDK de tercers que realitza la funció de *middleware* entre el dispositiu i el sistema. El seu software proporciona tècniques molt rellevants per NUI com és el *skeleton tracking*, el reconeixement de gestos, etc. Està desenvolupat per l'empresa 3DiVi Inc. i proporciona compatibilitat amb OpenNI. Això implica que sigui multi llenguatge i multiplataforma. Les característiques més rellevants que hi podem trobar són:

- Seguiment esquelètic del cos (19 articulacions) (veure figura 2.6)
- Reconeixement de gesticulacions (veure secció 2.2.5)
- Seguiment facial
- Un SDK multiplataforma per Windows, Linux i Android
- Suporta múltiples sensors (Kinect v1/v2, Orbbec Astra/Persee, etc.)
- Connectors o *plugins* per a diversos motors de joc

La part negativa en la utilització de Nuitrack és que no és 100% gratuït. És a dir, es disposa del SDK per a poder-lo utilitzar sense tenir-lo de pagar, però això implica que únicament es podrà executar en un temps màxim de 3 minuts per cada execució. Una vegada superat aquest temps, el *framework* deixarà de funcionar i no deixarà captar moviments. Tanmateix, no es podrà interaccionar amb l'aplicació mitjançant el *middleware*, altrament sí. En la pàgina web oficial de Nuitrack es pot adquirir la llicència per 39.99$ anuals o 99.99$ amb temps il·limitat[2].

Aquest SDK podria ser incorporat en molts àmbits de la investigació com podria ser per rehabilitació[2] per oferir gran compatibilitat, *smart home*, etc. A més també està bastant enfocat en la utilització en videojocs, ja que és possible combinar aquest sistema amb dispositius que permeten incorporar la realitat virtual (VR)[11]. Incorpora d'alguns exemples i tutorials on ensenya a realitzar petites demostracions per a provar el sistema[12].

[2]NUITRACK Pro for Windows. (2018, 27 de desembre). *Nuitrack Full Body Skeletal Tracking Software*. Data de consulta: 20:49, gener 14, 2019 des de `https://nuitrack.com/#pricing`.

### 2.2.2 | Arquitectura

L'API de Nuitrack és una interfície síncrona i nativa de C++. Està basada en una capa *middleware* que és l'encarregada d'amagar les comunicacions amb el sensor 3D. L'arquitectura que comprèn Nuitrack SDK és la que apareix en la següent figura 2.1.

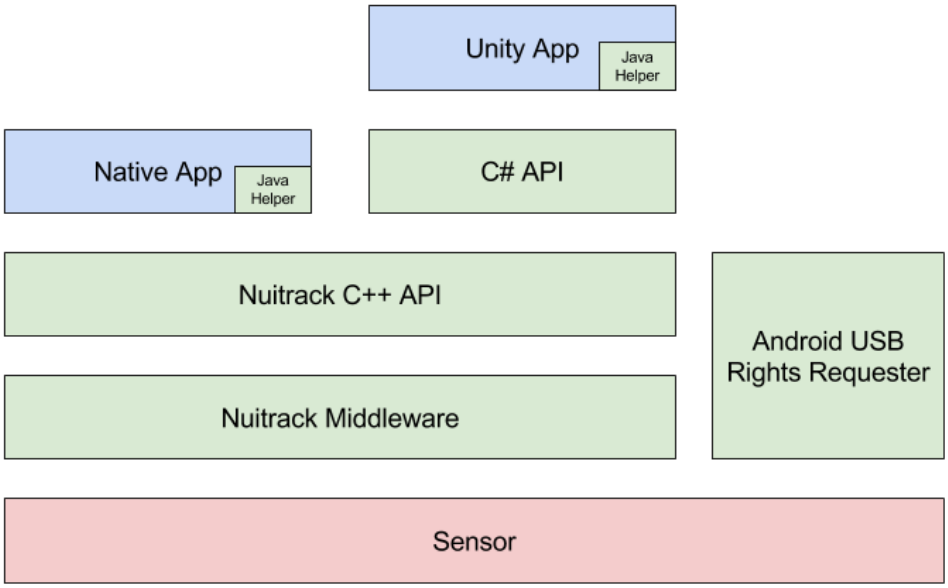



FIGURA 2.1: Arquitectura de Nuitrack SDK
Font: https://download.3divi.com

La seva arquitectura està composta per 5 capes les quals la més baixa de totes correspon a la capa hardware, és a dir, al sensor 3D que captarà els moviments de l'usuari. En la capa superior del sensor es troba la capa *middleware* de Nuitrack, que s'encarrega d'amagar les comunicacions entre els sensors compatibles per Nuitrack i l'API en C++. Per acabar, en les capes superiors es trobarà el plugins que permeten la compatibilitat entre el *middleware* i el software o llenguatge en el qual es programa l'aplicació[13].

### 2.2.3 | Preparacions generals

Abans de dur a terme qualsevol prova o execució de l'aplicació, cal assegurar-se que les condicions de l'entorn on es trobarà l'usuari que utilitzarà el sistema són les òptimes. Per tal de poder treure la màxima experiència d'usuari (UX) i el màxim rendiment en la part del dispositiu, és recomanable seguir els diferents consells que ofereix Nuitrack en la seva pàgina i que es veuran a continuació[14].

### Posició del sensor i entorn de l'usuari

Tant la posició del sensor com l'entorn d'actuació de l'usuari, és de vital importància complir una sèrie de requisits a l'hora d'utilitzar el sistema. Una mala posició del sensor o un entorn d'actuació incorrecte pot provocar que l'aplicació mostri els resultats o el *feedback* incorrectament i tingui problemes de detecció com els que apareixen en la figura 2.5. Així doncs, es recomana unes mesures estàndard per fer funcionar el sistema de manera correcta i sense problemes (vegeu figura 2.2).

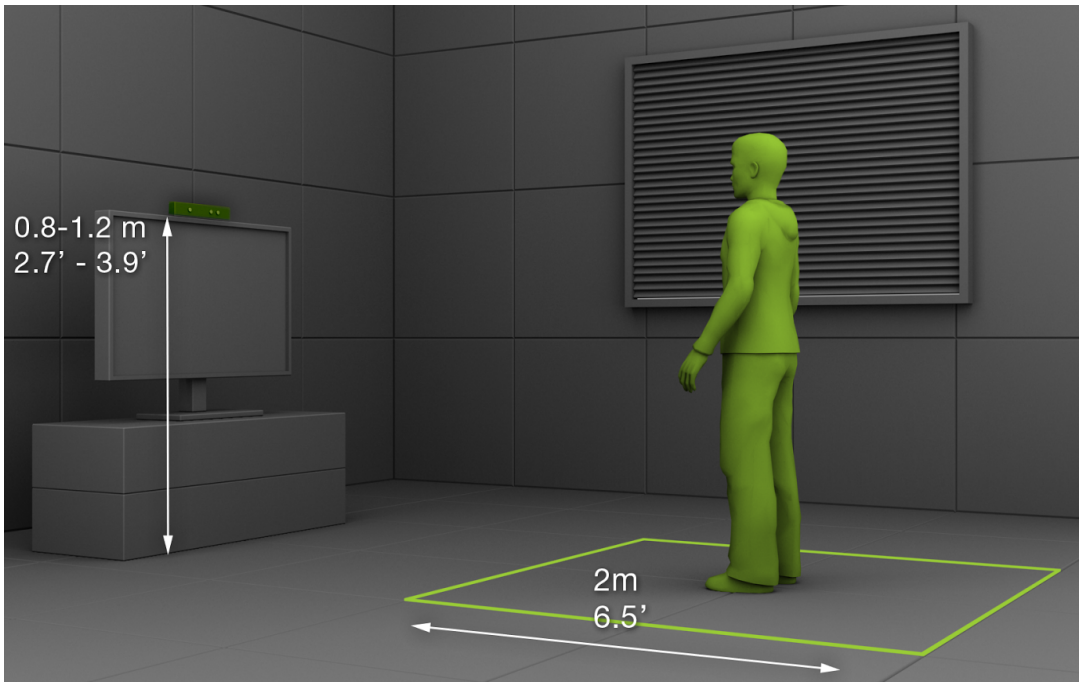



FIGURA 2.2: Posició del sensor i entorn de l'usuari
Font: https://download.3divi.com

### Condicions lumíniques

Per tenir una bona detecció per part del sensor que s'encarregarà de captar els moviments de l'usuari, és recomanable que la llum del sol no entri directament en el camp d'actuació de l'usuari (i menys al de la càmera) tal com s'indica en la figura 2.3, ja que això dificultarà la detecció de l'usuari i afectarà el correcte funcionament del sistema. Si és possible, il·luminar l'entorn amb llum natural de l'exterior exempts de rajos solars o bé, per mitjà de llums d'interior que també encaixarien amb les recomanacions.

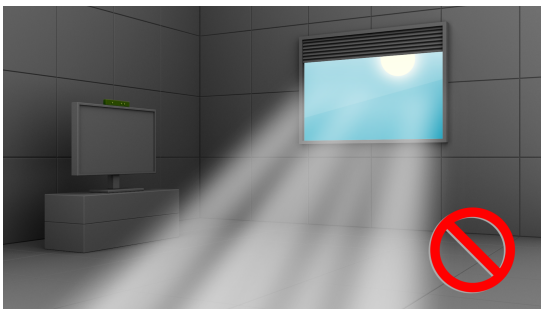


FIGURA 2.3: Prohibició de l'entrada de rajos solars en l'àrea d'actuació
Font: https://download.3divi.com

### Múltiples jugadors en la mateixa àrea d'interacció

En el cas que hi hagi 2 o més usuaris en l'àrea d'actuació del sensor, caldrà haver-hi un marge entre jugadors per tal de detectar les dues persones correctament. La superposició d'un usuari es pot veure afectat en la detecció i seguiment del sensor.

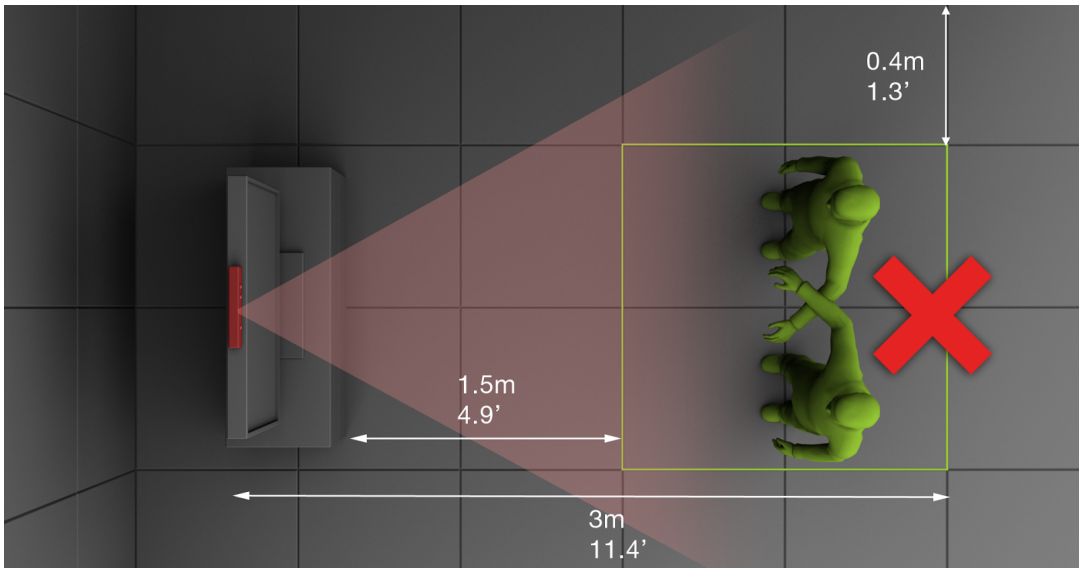



FIGURA 2.4: Superposició entre usuaris en l'àrea d'actuació
Font: https://download.3divi.com

Encara que els desenvolupadors de Nuitrack no ho especifiquen, el *middleware* és capaç de detectar fins a un total de 6 usuaris dins de l'àrea d'acció, encara que el principal problema podria ser l'àrea d'acció, ja que seria necessari que fos molt amplia per poder suportar tants usuaris al mateix temps.

### Visualització de l'usuari en la UI

És una recomanació que ve lligada amb la primera que s'ha explicat de *posició del sensor i entorn de l'usuari*. És important si no es vol tenir problemes de visualització en la pantalla, ja que es recomana com a mínim, que es vegi el cos humà des dels genolls, fins a tocar la punta del dit amb els braços aixecats el màxim possible. Per altra banda, la visualització dels peus no és rellevant en aquest projecte, ja que la interacció sempre es realitzarà amb els braços i les mans, la resta del cos és menyspreable.

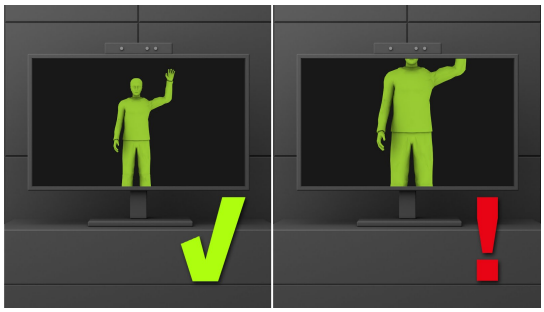


FIGURA 2.5: Posició del jugador en pantalla
Font: https://download.3divi.com

### 2.2.4 | Seguiment esquelètic del cos

El seguiment esquelètic o *skeleton tracking* és una tècnica que tracta el processament de la informació sobre la imatge i que estableix les posicions de les articulacions de l'esquelet en la forma humana. És capaç de detectar l'usuari en la imatge per la seva forma i assignar les diferents articulacions (cap, mans, peus, etc.) segons la ubicació de l'articulació en la imatge. Aquesta metodologia és capaç de calcular la posició de cada articulació en els eixos $x$, $y$ i $z$ de l'entorn de l'usuari[15].

El seguiment esquelètic de Nuitrack és capaç de seguir fins a sis usuaris, encara que per defecte sol estigui activat per dos. Per cada imatge que processa el SDK de Nuitrack, és capaç de generar fins a 19 articulacions en tot l'esquelet de l'ésser humà (vegeu figura 2.6).

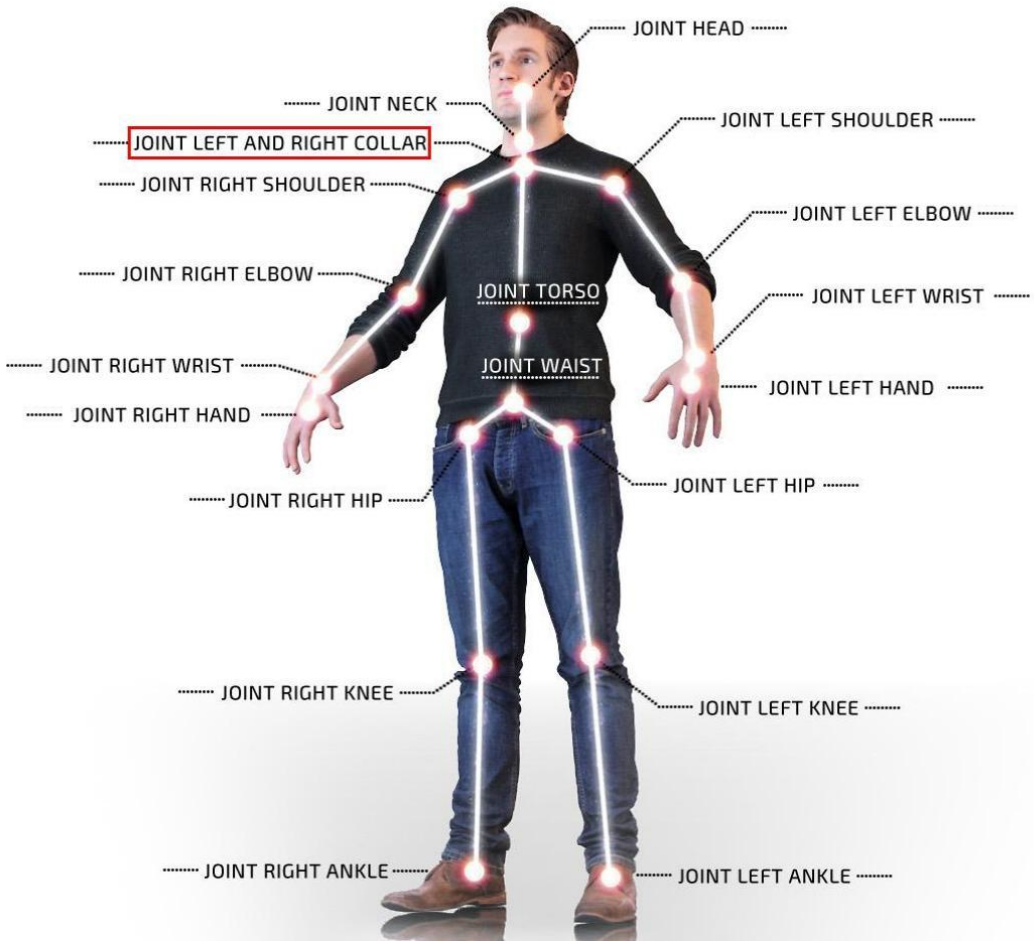



FIGURA 2.6: Articulacions detectades pel SDK de Nuitrack
Font: http://download.3divi.com/

Per ara, no s'ha especificat la possibilitat d'agregació de noves articulacions, això sí, s'han quedat obsoletes les articulacions dels dits a causa de les noves versions de Nuitrack que han aparegut, segurament per la imprecisió en el seguiment que es podia generar.

### 2.2.5 | Reconeixement de gestos

El reconeixement de gestos o *gesture recognition* és una tècnica que interpreta els moviments de l'usuari mitjançant algorismes matemàtics. És un mecanisme que serveix per provocar *feedback* al sistema i normalment vénen originats mitjançant les extremitats o en alguns casos el cap.

En Nuitrack s'ha incorporat aquesta característica i permet detectar els gestos que vénen inclosos en el SDK, no permet incloure nous gestos per donar més interacció amb el sistema. Així doncs, els gestos que inclou són els següents:

- **Clic** o *click*. És l'acció que és dur a terme mitjançant les mans de l'usuari quan realitza l'acció d'obrir i tancar la mà.
- **Pressió** o *push*. És l'acció de realitzar la simulació a una pressió d'un botó, és a dir, apropar o allunyar la mà de la imatge del sensor.
- **Saludar** o *waving*. És l'acció de saludar fent anar l'extremitat superior d'un costat a l'altre de la imatge.
- **Lliscar** o *swipe*. És l'acció de desplaçar el braç cap a una direcció (la unió de dues d'elles podrien fer el *waving*). Les variants que existeixen són:
  - **Esquerra** o *left*. Lliscar cap a l'esquerra davant la càmera.
  - **Dreta** o *right*. Lliscar cap a la dreta davant la càmera.
  - **Amunt** o *up*. Lliscar cap amunt davant la càmera.
  - **Avall** o *down*. Lliscar cap avall davant la càmera.

## 2.3 | Motor de joc: Unity

En el desenvolupament del projecte és necessari un IDE capaç de poder interactuar amb el *middleware* explicat en l'anterior secció 2.2, on s'haurà de crear tot el vincle amb el motor de joc i que sigui capaç de poder-se comunicar amb el sensor i d'actuar en conseqüència. Per tant, l'IDE que més s'adapta als requisits del projecte és el motor de joc d'Unity 3D.

### 2.3.1 | Què és Unity 3D?

Unity 3D de l'empresa Unity Technologies és un potent motor de jocs utilitzat mundialment avui en dia, en gran part, per la creació de jocs, animacions i aplicacions. És multiplataforma i treballa tant en 2D com en 3D. Per altra banda, Unity és un software parcialment gratuït i propietari, on es pot executar en Windows, Linux i Mac OS X. De les característiques més rellevants que es poden destacar per la importància en aquest projecte són:

- **Gran contingut en l'Asset Store**. Mercat de *assets*, textures, materials i de tota classe d'elements, similar a una App Store, que usuaris d'Unity poden posar a la venda (també hi ha contingut gratuït), on també és possible descarregar contingut i incloure'l al projecte.

- **És compatible amb Nuitrack SDK**. El SDK de Nuitrack disposa d'un paquet de *assets*, que és un connector que permetrà la comunicació entre la capa de software Unity i la capa del *middleware* que es comunicarà amb la càmera Orbbec.
- **Te suport de compilació amb diferents tipus de plataformes**. Com s'observa en la següent figura 2.7, Unity 3D permet compilar projectes per a una gran varietat de dispositius i sistemes operatius.
- **Motors de física**. Gràcies amb aquests motors de física, és possible realitzar projectes molt realistes i d'alt rendiment al voltant de l'ús de físiques en el joc.

Unity utilitza el llenguatge MonoDevelop[3] per fer els scripts necessaris per definir el comportament de l'aplicació. Principalment el llenguatge de programació de MonoDevelop és C#, però també suporta .NET, Java i Python.

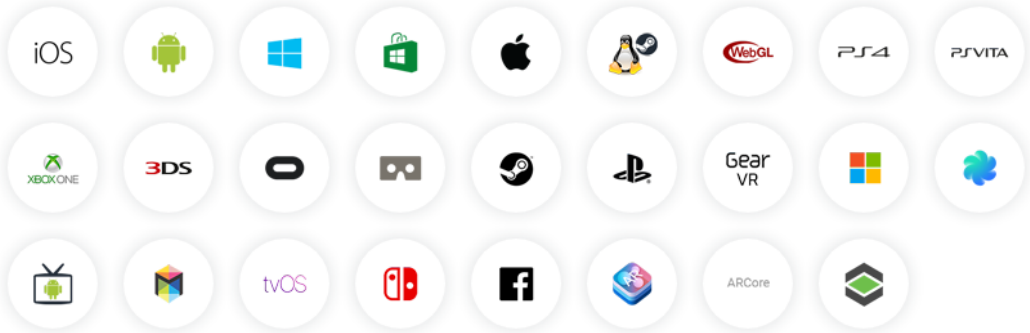



FIGURA 2.7: Icones de les plataformes suportades per Unity 3D
Font: https://unity3d.com/es/unity

El motiu pel qual s'ha escollit el motor de joc Unity 3D és per la quantitat de material de suport i ajuda que ofereix. En la web de 3Divi també existeix tutorials i ajudes per treballar en aquest motor de joc. Tot material i referències que es puguin aprofitar servirà de molta ajuda pel desenvolupament del sistema, ja que les referències trobades són mínimes en aquest àmbit.

### 2.3.2 | Integració de Nuitrack$^{TM}$SDK

La utilització d'aquest *framework* no serà compatible directament amb Unity 3D, per això serà necessari disposar de connectors per la utilització de Nuitrack SDK a Unity 3D. El funcionament del connector és senzill i és tal com apareix en la següent figura 2.8, aquest està compost per una llibreria d'enllaços dinàmics (DLL) programada en C++ i que actua com intermediària entre l'script en Unity i el SDK de Nuitrack. Està compost d'un script d'Unity que realitza la funció d'interfície (programada en C#) i que s'encarrega d'invocar funcions en la DLL des del codi d'Unity 3D. Tot això, és similar a integrar Microsoft Kinect SDK que fa referència al sensor Kinect [16].

[3]MonoDevelop. (2018, 8 de gener). *Wikipedia, La enciclopedia libre*. Data de consulta: 10:03, gener 18, 2019 des de `https://es.wikipedia.org/w/index.php?title=MonoDevelop&oldid=104770774`.

La funció principal del connector és transportar les crides realitzades en la interfície d'Unity 3D (script C#) fins al SDK de Nuitrack. Aquest SDK realitzarà la funció d'intermediari (*middleware*) entre el software d'Unity 3D i la càmera 3D.

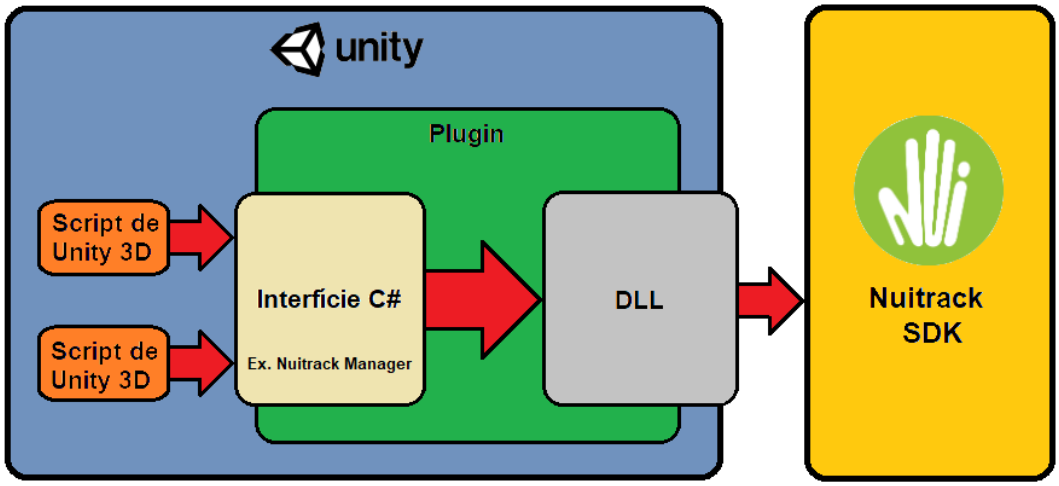



FIGURA 2.8: Arquitectura del plugin de Nuitrack en Unity
Font: https://www.researchgate.net (modificada)

## 2.4 | Requeriments i eines addicionals

Per la instal·lació d'Unity 3D, explicada en l'anterior secció 2.3, serà necessari que l'ordinador en el qual es desenvoluparà el projecte, tingui unes prestacions de gamma mitjana. És molt recomanable l'ús d'una targeta gràfica dedicada per poder-hi renderitzar les textures el projecte que es desenvoluparà. Tanmateix, serà obligatori crear-se un compte d'Unity (Unity ID) per poder utilitzar Unity 3D i no serà obligatori utilitzar la versió professional, ja que tot el desenvolupament que es mostrarà a partir del capítol 4, s'ha desenvolupat en la versió gratuïta.

Com a eines addicionals que formaran part del projecte i que únicament s'utilitzaran en el procés de desenvolupament són, per una part, Visual Studio 2017 de l'empresa Microsoft, ja que de manera opcional s'instal·la amb Unity 3D i conté eines que faciliten la programació de scripts. No és obligatori instal·lar-lo, ja que existeixen altres eines compatibles amb Unity que podrien reemplaçar aquest software si fos necessari.

Per altra banda, la segona i última eina addicional en aquest projecte serà un control de versions o repositori en línia com GitHub de l'empresa Microsoft. S'ha escollit aquest tipus de control de versions, ja que es guarda en la xarxa i així s'està exempt de qualsevol pèrdua d'informació i es té un millor control del projecte. El repositori es crearà de forma privada evitant així el codi obert i únicament els integrants del projecte,els directors i codirectors podran veure i modificar el contingut. Aquest servei es gestionarà mitjançant Source Tree[4] de l'empresa Atlassian.

[4]Sourcetree. https://www.sourcetreeapp.com/

# 3 Planificació del desenvolupament

## 3.1 | Decisions del projecte

A continuació, s'expliquen les decisions que s'han dut a terme abans de realitzar el desenvolupament del sistema per tal de millorar els possibles aspectes del prototip, ja sigui en la part de disseny, com en la part de desenvolupament. Cal fer èmfasi, que les decisions de projecte que es prendran no seran del projecte en general, sinó que seran totes elles, sobre el mòdul de *manipulació de l'objecte 3D en un entorn tridimensional*, on s'ha explicat prèviament en la descripció del projecte.

### 3.1.1 | Decisions d'implementació

Les primeres decisions que s'han pres sobre el projecte fan referència a la forma d'implementació del sistema. Totes elles estan relacionades en la forma i en les eines en què es durà a terme el desenvolupament del mòdul.

#### Agregació d'un *middleware*

La primera decisió important que s'ha pres, fa referència a l'agregació d'un software de tercers. Un *framework* que proporciona compatibilitat amb múltiples sensors, com ja s'ha explicat en la secció 2.2 i que en gran mesura, s'optarà per aquesta opció perquè proporciona suport per poder realitzar diversos tutorials per veure el potencial d'aquest *middleware*[17][18].

Els motius pels quals no s'ha implementat el projecte amb Astra SDK, que és proporcionada gratuïtament per Orbbec, ve donat perquè el SDK de Nuitrack ofereix més avantatges en termes de compatibilitat, característiques i ajuda, que no el que proporciona l'empresa Orbbec en la seva pàgina. Altrament, utilitzar Astra SDK implicarà la pèrdua de diversos avantatges per la part de Nuitrack com el seguiment de l'esquelet, ja que serà de vital importància per aquest tipus de projecte. Astra SDK també incorporà el seguiment d'esquelet però, no serà sempre gratuït, ja que a partir del 30 de juny serà considerat com una extensió del SDK i per tant, caldrà

posar-se en contacte amb l'empresa per adquirir aquesta característica per mitjà d'alguna quota mensual[19].

Al llarg del desenvolupament del projecte, Orbbec ha proporcionat una alternativa diferent del seu SDK com és el software de tercers Gestoos (veure Secció 2.1), però no s'ha aconseguit obtenir el SDK per provar-lo. En conclusió, es prescindirà de la utilització del software tant d'Orbbec com de Gestoos, a causa de les conseqüències que comporta cadascun d'ells.

#### Eina de desenvolupament

Per l'elecció de l'eina en què es durà a terme el desenvolupament del projecte, s'ha tingut en compte si s'agregava el *middleware* o no. També, a més, si és dur a terme amb el *middleware* caldrà tenir present que aquest ofereixi compatibilitat amb Nuitrack per poder fer ús dels plugins i configurar-lo de manera fàcil.

Nuitrack dóna suport tant a Unity 3D, com a Unreal Engine[1] de l'empresa Epic Games. També disposa de diversa documentació i tutorials, com s'ha explicat anteriorment, per part dels dos IDE.

Entre les dues opcions exposades, s'ha escollit Unity 3D, ja que de tots dos, és el més conegut, el que més documentació disposa fins al moment i el que més facilitat d'aprenentatge ofereix[20]. Per altra banda, la disposició de plugins compatibles amb Nuitrack facilita així la comunicació i programació amb el sensor.

### 3.1.2 | Decisions de disseny

A continuació s'explicaran les decisions inicials de disseny que s'han pres sobre el prototip, el qual s'engloba diversos aspectes visuals de l'aplicació, així com alguns aspectes de la seva funcionalitat.

Encara que no sigui un joc, serà necessari per a l'experiència de l'usuari la consideració diversos aspectes o regles que seran de vital importància per satisfer aquesta UX. Tot això, amb els recursos dels quals es disposa. Aquests aspectes són:

- **Mobilitat limitada**. L'usuari que estarà davant del sensor, únicament utilitzarà les extremitats superiors per utilitzar l'aplicació, la resta del cos és menyspreable. La idea principal és poder modelar la figura en 3D únicament amb les dues extremitats superiors, per dur a terme aquest objectiu, no és necessari que l'avatar del jugador es mogui de lloc.
- **Interacció amb l'entorn senzill**. La interacció, a part d'interactuar amb la figura que s'ha de modelar, en un primer moment no es podrà dur a terme. L'entorn fora de l'àrea d'acció de l'avatar en aquest projecte també és menyspreable.

[1]Unreal Engine. (2019, 5 de gener). *Wikipedia, La enciclopedia libre*. Data de consulta: 17:23, gener 21, 2019 des de https://es.wikipedia.org/w/index.php?title=Unreal_Engine&oldid=113075711.

- **NUI en el 100% de l'aplicació**. La interacció amb el sensor serà fonamental per poder utilitzar l'aplicació. A part d'utilitzar el sensor per realitzar el modelatge, també s'utilitzarà el sensor per navegar pels menús del sistema, però d'això s'encarregarà el mòdul d'*implementació de les interfícies d'usuari*.
- **Modelatge limitat**. S'ha de tenir present que Nuitrack SDK incorpora el seguiment de l'esquelet i el reconeixement de gesticulacions per tant, detecta els moviments amb les extremitats i els possibles gestos. Independentment, associar diversos gestos o moviments en una interacció natural sobre un objecte, com podria ser modelar-lo, no és fàcil i menys si el sensor no és 100% precís amb tots els moviments.

## 3.2 | Determinació de les *issues*

En la fase inicial del projecte, es desenvoluparan totes les *issues* que es duran a terme a l'hora de desenvolupar el projecte, tenint en compte les decisions que s'han pres en la secció 3.1. L'aparició d'aquestes, anirà en funció del desenvolupament del projecte, ja que a mesura que s'avança, això pot provocar noves *issues*, sigui en forma d'errors o en forma de noves implementacions/funcionalitats. El nombre d'iteracions que s'han tingut en compte a les *issues* és aproximat, el nombre d'iteracions dependrà del nombre d'entrebancs que vagin sorgint en el desenvolupament.

Tenint en compte els objectius establerts en el preàmbul d'aquesta memòria i les decisions d'implementació i de disseny, s'han identificat les següents *Issues* per iniciar el projecte:

– *Nuitrack configuration*: Configurar el sistema per poder utilitzar la càmera 3D des del sistema operatiu de Windows. Es comprovarà que es rep *feedback* per part de la càmera en el SO.

– *Package managment and additions*: Instal·lar els paquets necessaris per a la instal·lació del plugin de Nuitrack a l'IDE d'Unity 3D.

– *Main scene: Iteration 1*: Crear l'escena principal del projecte:

  1. Afegir el plugin de Nuitrack a l'escena.
  2. Crear l'avatar i implementar el seguiment d'esquelet.
  3. Crear l'entorn de l'escena.
  4. Aplicar la física i malles de col·lisió

– *Main scene: Iteration 2*: Continuar amb el desenvolupament del projecte:

  1. Afegir un *feedback* a la figura quan hi hagi contacte.
  2. Moure la figura a partir del contacte produït per l'avatar.

– *Main scene: Iteration 3*: Finalitzar el desenvolupament del projecte:

  1. Implementar el reconeixement de gestos.
  2. Agafar la figura a manipular.
  3. Deformar/modelar la figura a manipular.

– *Documentation*: Realitzar la corresponent memòria del projecte i els respectius informes de seguiment durant el període del treball de final de grau.

En el procés de desenvolupament del prototip, segueix una metodologia de programació extrema (XP) on únicament s'acabarà la iteració quan hi hagi un contratemps que no tingui cap solució. A l'inici de la nova iteració s'hauran d'implementar els objectius de la iteració anterior més els objectius de la iteració actual, ja que es tornarà a començar de nou.

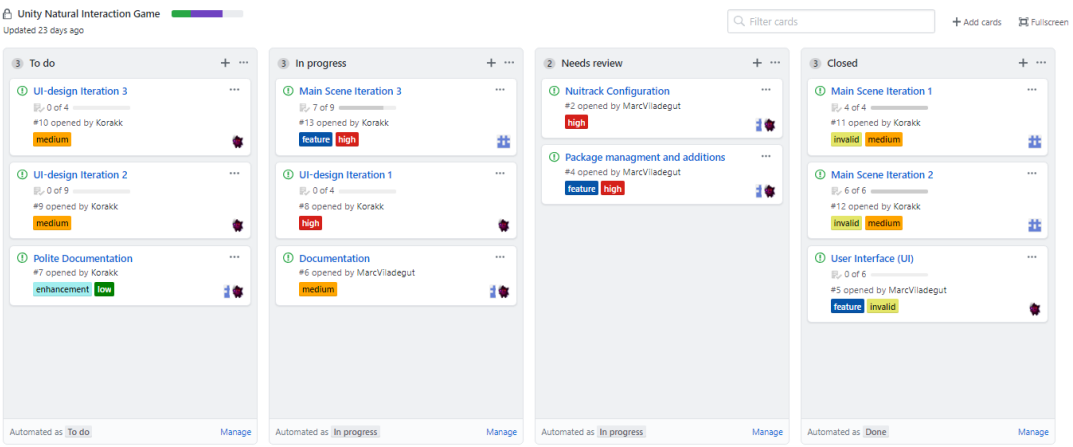



FIGURA 3.1: Exemple de kanban del projecte en fase avançada
Font: https://github.com/MarcViladegut/TFG/projects/1

S'utilitzarà una eina per la gestió de les *issues* anomenada Kanban[2] i que forma part de les metodologies Agile pel desenvolupament de software. Aquestes *issues* es dividiran en funció del seu estat actual en el projecte, així es tindrà un millor control i vista del projecte. En la següent figura 3.1 es pot observar el kanban en fase avançada del projecte en la plataforma Github.

## 3.3 | Desenvolupament i estructura general del projecte

Pel desenvolupament del projecte s'implementarà la programació extrema tal com s'ha esmentat en l'anterior secció 3.2, aquest desenvolupament estarà basat en iteracions, on cada iteració suposarà una nova versió millorada del projecte i seran finalitzades únicament quan s'hagin completat tots els objectius establerts per aquella iteració, o bé quan no hi hagi solució possible per continuar amb el desenvolupament. Cada iteració estarà formada per 3 fases que seran: anàlisis, disseny, desenvolupament.

En la fase d'anàlisis es realitzaran els anàlisis de requisits al principi de cada iteració per a determinar els objectius que s'afegiran i/o es modificaran. Respecte a la fase de disseny, s'aprofitarà per a millorar l'estructura i codificació del codi sense

[2]Kanban. https://kanbantool.com/es/metodologia-kanban

alterar la funcionalitat. En la fase de desenvolupament, únicament es desenvoluparà el projecte completant les diferents *issues* que existeixen. Per acabar, s'afegirà un camp addicional en la memòria anomenat *dificultats trobades*. La idea és dur a terme una petita anàlisi amb les dificultats o problemes que s'han anat trobant al llarg del desenvolupament de la iteració en qüestió i posteriorment, el motiu, si s'escau, del canvi o finalització de la iteració en curs.

Pel desenvolupament general del projecte basat en iteracions, el projecte es dividirà en dos mòduls principals els quals es desenvoluparan de forma totalment paral·lela entre els diferents membres. La primera part s'encarregarà de manipular l'objecte 3D amb la interacció basada en moviment i tot el que això comporta. La segona i última part, s'encarregarà de les interfícies d'usuari que tindrà el sistema i en la seva forma d'interacció, on se centrarà completament en un disseny centrat en l'usuari.

En la següent figura 3.2 es podrà observar el desenvolupament general previst i explicat anteriorment sobre el projecte. Es pot observar que en cada iteració la part d'interfícies d'usuari disposa d'una fase més per iteració, ja que en aquesta es durà a terme el *focus group*.

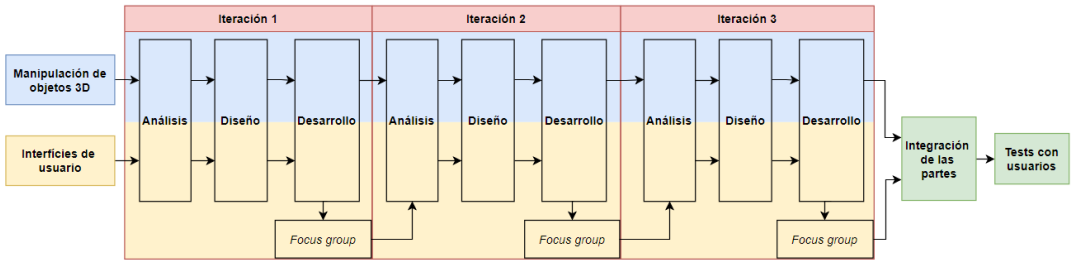



FIGURA 3.2: Metodologia del desenvolupament del projecte
Font: Creació pròpia

El nombre d'iteracions de la figura 3.2 és orientatiu, aquest pot variar en el nombre en funció de si apareixen noves *issues* o cal incrementar el nombre d'iteracions a causa de diferents motius en el desenvolupament. Pot ser que les diferents parts del projecte treballin en iteracions diferents, ja que es duran a terme totes dues de forma totalment independent i únicament s'integraran totes dues al finalitzar les iteracions.

En la integració, s'ajuntarà les dues parts al final del desenvolupament i es passarà un test conjunt amb els usuaris per obtenir les valoracions i treure'n les conclusions finals al projecte. Una vegada superat aquest pas, es donarà el projecte per conclòs.

# 4

# Desenvolupament: Manipulació d'objectes 3D

En aquesta part del mòdul *manipulació de l'objecte 3D en un entorn tridimensional* es durà a terme tot el desenvolupament sobre aquest per arribar a l'objectiu principal d'arribar a manipular una figura 3D tant en la seva posició, com en la seva forma. Aquest procés es durà a terme a partir de les diverses tècniques necessàries que incorpora Nuitrack per tal de poder dur a terme una interacció natural.

El desenvolupament d'aquest mòdul es durà a terme per mitjà d'una variant de la metodologia de *extreme programming* basat en iteracions. Com s'ha explicat anteriorment, on el canvi d'iteració suposa que no hi ha camí possible per seguir amb la iteració i per tant, s'abordarà el desenvolupament des d'un altre camí i en una nova iteració.

## 4.1 | Primera iteració

En aquesta primera iteració s'intentarà crear tota l'escena que emmagatzemarà el nucli central del videojoc. La idea és com a mínim aconseguir la interacció o almenys arribar a provocar el *feedback* al videojoc a partir dels moviments produïts dins de l'àrea d'acció de la càmera 3D.

### 4.1.1 | Anàlisis

En aquesta iteració s'especificaran els següents requeriments per dur a terme en aquesta fase inicial del projecte:

- Afegir les extensions de Nuitrack a escena
- Crear l'avatar i implementar el seguiment d'esquelet
- Crear l'entorn de l'escena on transcorrerà tota la interacció

- Aplicar les físiques i malles de col·lisió als objectes pertinents

Tots aquests objectius estan especificats en la *issue* anomenada *"Main scene: Iteration 1"* de la secció 3.2. Per tant el principal objectiu d'aquesta iteració serà completar tots els requeriments de manera satisfactòria.

### 4.1.2 | Disseny

Respecte al disseny d'aquesta iteració, s'ha plantejat des d'un bon principi d'arribar en un desenvolupament similar tal com es mostra en la següent figura 4.1. Posteriorment, es donarà moviment a l'avatar que surt a la figura mitjançant la tècnica del seguiment de l'esquelet per fer-ho possible.

FIGURA 4.1: Disseny de l'escena on apareix l'avatar que serà controlat per l'usuari mitjançant la interacció basada en moviment
Font: http://download.3divi.com

Respecte a l'entorn, es donarà una temàtica al videojoc mitjançant els *assets* gratuïts que hi hagi en l'Asset Store d'Unity 3D. Una primera idea serà posar un entorn tancat envoltant a l'avatar i posar-hi diferents objectes, on no es podrà interactuar amb ells, i que únicament formaran part de la decoració.

Respecte a les físiques, s'aplicarà gravetat en aquells objectes que sigui estricament necessari dóna'ls-hi aquesta propietat i sobretot, a l'objecte que es modelarà. En un primer moment, tots els objectes que formin part de l'entorn seran menyspreables. El mateix passarà amb les malles de col·lisió.

### 4.1.3 | Desenvolupament

L'inici del desenvolupament començarà amb la creació de l'escena en Unity 3D que emmagatzemarà tot el nucli central de l'aplicació. Per la creació d'aquesta escena caldrà crear prèviament el projecte que emmagatzemarà el mòdul principal i que s'ha explicat en la secció 3.3.

#### Creació del projecte

La creació del projecte es durà a terme tal com apareix en la següent figura 4.2, es pot observar la configuració inicial que se li donarà al projecte en Unity abans de crear-lo. Els camps que es destacaran són el *Template* i *Add Asset Package*.

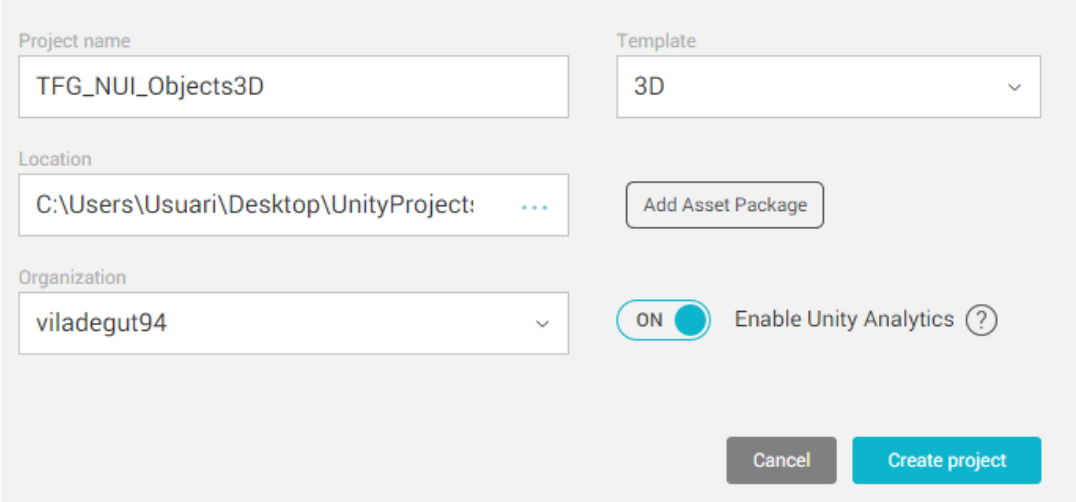



FIGURA 4.2: Interfície de creació d'un nou projecte en Unity 3D
Font: Creació pròpia

En el camp *Template* serà imprescindible seleccionar 3D, ja que des d'un bon inici, Unity 3D crearà un entorn tridimensional, amb els eixos $x$, $y$ i $z$. El camp de *Add Asset Package* serà menyspreable en aquesta ocasió, ja que inicialment no s'agregaran paquets de *assets* en la mateixa creació del projecte, sinó que es realitzarà, una vegada creat el projecte, una cerca per la botiga d'Unity (Asset Store). El fet d'agregar un paquet en l'inici del projecte, implica que prèviament s'ha hagut de cercar i afegir a la biblioteca d'*assets*, des de la botiga d'Unity 3D. Una vegada s'han introduït la corresponen informació en tots els camps, es procedirà a la creació del projecte que emmagatzemarà el mòdul.

#### Agregació d'objectes

Amb la creació del projecte en la secció 4.1.3, el següent pas serà importar els *assets* que aportaran un disseny i un entorn per poder donar realisme i temàtica al prototip. La incorporació d'aquests paquets facilita bastant la feina, ja que no caldrà dur a terme un procés de disseny per crear els elements que formaran part del prototip.

Per agregar els *assets* en una escena, existeixen dues maneres de fer-ho:

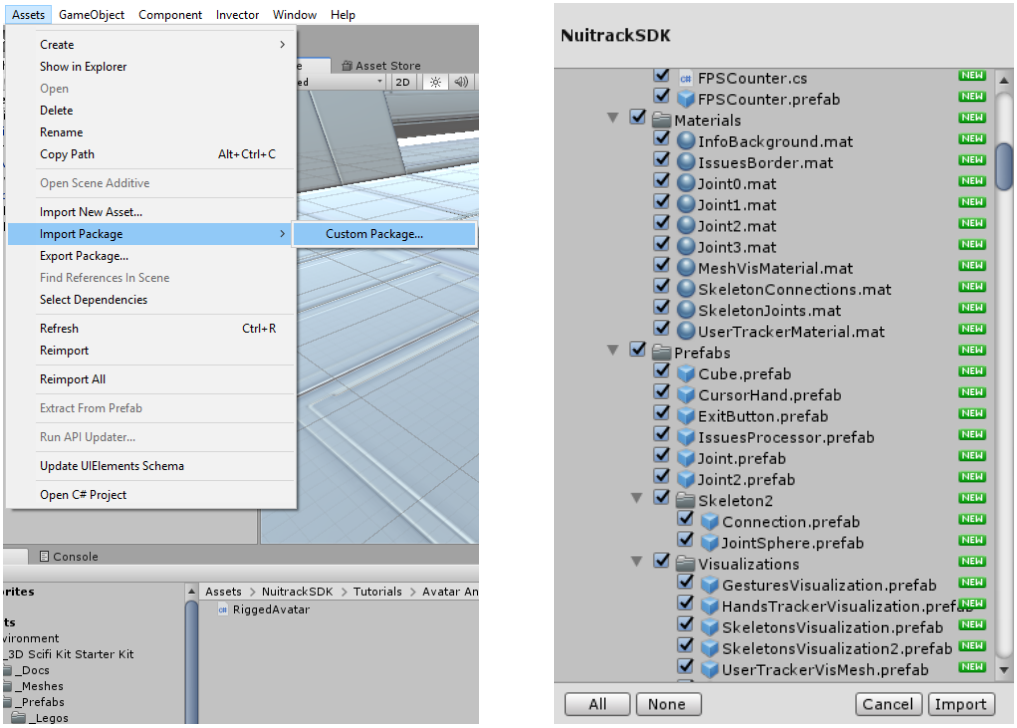
- **Des de l'Asset Store**. Descarregant paquets d'objectes des de la botiga virtual d'Unity i importar-los al projecte per a poder-los utilitzar dins de les escenes.
- **Des de paquets externs**. Són paquets personalitzats del tipus *unitypackage* i el seu contingut no deixa de ser res més que textures, animacions, scripts, etc.

Per la creació de l'entorn i l'avatar s'utilitzaran els *assets* importats de l'Asset Store. Els paquets que s'han escollit i que formaran part del prototip seran els següents:

– **3D Scifi Kit Starter Kit**[21]. Paquet de components del joc que conté textures, objectes, etc. S'utilitzarà per a crear l'entorn de l'escena principal on es desenvoluparà tota la interacció.

– **Third Person Controller - Basic Locomotion FREE**[22]. Paquet de components i prefabs[1] que conté el disseny d'un robot i que serà l'encarregat d'imitar els moviments que faci l'usuari i manipular la figura dins de l'escena.

Serà necessària la importació de Nuitrack en aquest punt per tal de crear la comunicació entre l'IDE i la càmera Orbbec, ja que de no importar-se, la càmera no transmetrà cap dada al programa. Per la importació, el SDK de Nuitrack es podrà realitzar de les dues formes esmentades anteriorment, és a dir, importar-lo de la botiga d'Unity[23], o bé de manera externa. En aquest cas, s'importarà de manera externa des de la web de Nuitrack[2].




(A) Importació d'un paquet personalitzat (B) Importació d'objectes dins de l'*asset*

FIGURA 4.3: Importació de manera externa a Unity

[1]Prefabs. (2018, 31 de juliol). *Unity | Documentation*. Data de consulta: 16:35, gener 22, 2019 des de https://docs.unity3d.com/es/current/Manual/Prefabs.html.

[2]Nuitrack API. https://nuitrack.com/#rec54893368.

Una vegada descarregat el SDK, caldrà instal·lar-lo a Unity 3D per mitjà d'un paquet, també anomenat *unitypackage*, que és el tipus de l'arxiu, i que es trobarà dins del SDK de Nuitrack descarregat prèviament. Per la instal·lació, serà necessari importar-lo des de dins d'Unity en el menú de "Assets" (vegeu Figura 4.3a). Tot seguit, s'obrirà a Unity una finestra mostrant els components que es voldran importar d'aquell paquet. En aquest cas, s'importaran tots els components per tal de no crear conflictes en la compilació del projecte (vegeu Figura 4.3b), encara que no és necessari la importació de tots ells.

Una vegada s'han importat aquests 3 paquets d'objectes al projecte, es podrà començar a treballar amb la implementació tant del disseny de l'avatar, com del seguiment de l'esquelet d'aquest. Posteriorment, es podrà realitzar la creació de l'entorn, però això no serà tan rellevant.

### Inicialització del SDK de Nuitrack

En el SDK de Nuitrack importat prèviament, existeixen diversos *prefabs* els quals destacarem un d'ells anomenat *NuitrackScripts* que conté els *scripts* necessaris per interactuar amb Nuitrack des de l'entorn d'Unity. Aquest objecte està compost per diversos *scripts* que són:

- **Nuitrack Manager**. S'escull les accions que es vol que detecti Nuitrack pel sensor i s'activaran per mitjà d'un *checkbox*. Les accions poden ser:
  - Depth Module On
  - Color Module On
  - User Tracker Module On
  - Skeleton Tracker Module On
  - Gesture Recognizer Module On
  - Hands Tracker Module On
- **Current User Tracker**. *Script* necessari per al seguiment de l'esquelet de l'usuari en Nuitrack.
- **T Pose Calibration**. S'implementa un calibratge previ a l'execució de l'aplicació per verificar que el sensor detecta els moviments correctament. Únicament funciona amb els dispositius TVico.
- **Calibration Info**. Conté paràmetres addicionals pel calibratge en Nuitrack.

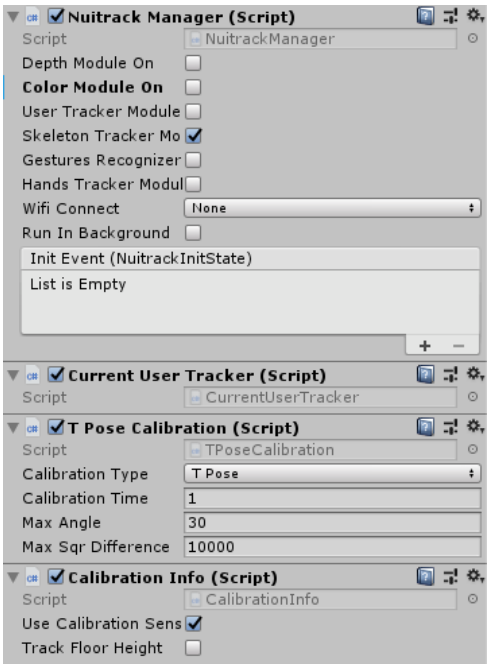



FIGURA 4.4: Components que componen el *prefab* de Nuitrack en Unity
Font: Creació pròpia

La idea és deixar aquest *prefab* en l'escena, visualment serà invisible, i tot seguit marcar l'opció del seguiment d'esquelet (*Skeleton Tracker Module On*). Posteriorment, Nuitrack hauria de ser capaç de detectar si un usuari està davant del sensor, detectar les diferents articulacions que disposa i seguir els moviments que realitza l'usuari.

**Disseny i estructura de l'avatar**

L'avatar utilitzat en aquest projecte serà el que representarà a l'usuari en l'entorn tridimensional del prototip. Aquest correspon a un paquet de *assets* importat des de la botiga d'Unity 3D (vegeu Secció 4.1.3), el qual conté un conjunt de *prefabs* per a poder crear l'avatar. El resultat d'afegir aquests components a l'escena genera com a resultat un avatar com el que apareix en la figura 4.5. Aquest *asset* és bastant complet, ja que permet treballar en l'àmbit d'articulacions i modificar-les segons els requeriments en cada context amb la màxima flexibilitat possible. També inclou diversos *gameobjects* per agregar més realisme i característiques com animacions, scripts, models d'avatar, textures, etc.

FIGURA 4.5: Avatar de l'escena principal del prototip
Font: Creació pròpia

Una vegada s'agrega el *prefab* de l'avatar a l'escena, apareixerà l'avatar amb els braços totalment oberts i adoptant una posició com de lletra T. Si s'observa la composició d'aquest *prefab*, es pot observar la similitud entre les articulacions de la figura 2.6 de la secció 2.2.4 i l'objecte importat en l'escena. Òbviament, no tindrà el mateix nombre exacte però sí que hi haurà moltes que podran tenir relació.

Caldrà enllaçar aquesta similitud entre els components que formen l'avatar i les articulacions que detecta Nuitrack. Cal tenir present, que no tots els avatars estan compostos pel mateix nombre d'objectes i articulacions, cada un dependrà del seu dissenyador. Per tant, es pot donar el cas que hi hagi avatars que siguin més o menys compatibles amb Nuitrack depenent del nombre d'articulacions que contingui cadascun d'ells. En un menor nombre d'articulacions en l'avatar, sempre es pot prescindir d'algunes articulacions que considerem menyspreables per la part de Nuitrack, en canvi si és superior, els moviments de les articulacions no seran del tot realistes/precisos possible.


FIGURA 4.6: Objectes que componen la figura de l'avatar
Font: Creació pròpia

Els components de l'avatar es poden observar en la següent figura 4.6. Aquests estan compostos per diversos objectes que a la vegada aquests també estan compostos per diversos objectes. La idea és emmagatzemar aquelles articulacions més petites com poden ser els dits de les mans, dins d'objectes que formen el conjunt de tots ells, com poden ser les mans. Partint que la cintura del personatge serà l'articulació arrel, aquesta estarà composta del tronc, la cuixa dreta i la cuixa esquerra i així successivament fins a arribar al

final de totes les extremitats. Cada element d'aquest que forma l'avatar serà una articulació que s'associarà amb les articulacions que s'obtinguin per part de Nuitrack. Una vegada s'han posat l'avatar en l'escena, serà important deixar els valors de la rotació i posició de l'avatar per defecte, per evitar així, càlculs erronis a l'hora de produir-se el seguiment de l'esquelet en l'avatar.

**Animació de l'avatar (*skeleton tracking*)**

Per poder donar-li moviments a l'avatar perquè es pugui moure com ho realitza l'usuari dins de l'àrea d'acció del sensor, es durà a terme mitjançant la tècnica que proporciona el SDK de Nuitrack anomenada *skeleton tracking* (vegeu secció 2.2.4).

En aquesta secció es tractarà d'associar les respostes sobre cada articulació que detecta el sensor originats per l'usuari i provocar el *feedback* corresponent sobre l'avatar d'Unity. Per això, serà necessari crear un vincle amb cadascuna de les articulacions detectades pel sensor i connectar-les amb les articulacions de l'avatar perquè imiti els moviments. Per fer-ho, serà necessari emprar *scripts* d'Unity. L'objectiu serà que l'avatar es mogui tal com ho realitzarà l'usuari que farà ús del prototip, com per exemple en la figura 4.7.

FIGURA 4.7: Moviment d'un avatar per mitjà del sensor (GIF)
Font: http://download.3divi.com/

S'ha d'anar amb compte amb les rotacions i posicions de les articulacions superiors que imitarà a l'usuari, ja que no serà el mateix que s'imiti els moviments amb efecte mirall que a l'invers. En aquest cas, l'*script* que es durà a terme serà de forma natural, és a dir, sense efecte mirall, ja que posteriorment se situarà la càmera en la direcció de l'avatar, simulant el prototip en una aproximació a un joc en primera persona.

Per dur-ho a terme, inicialment caldrà crear una classe que serà l'encarregada d'emmagatzemar diversos atributs imprescindibles per produir moviments, almenys en les dues extremitats de l'escena, aquesta classe s'anomenarà *ModelJoint.cs*. Doncs bé, en aquesta classe s'emmagatzemaran els atributs que s'explicaran a continuació:

- **bone**. Objecte del tipus *Transform*[3] que emmagatzemarà l'articulació de l'extremitat dins l'escena.
- **jointType**. Objecte del tipus *nuitrack.JointType*[4] que emmagatzemarà una articulació reconeguda per Nuitrack de les 19 possibles.
- **baseRotOffset**. Objecte del tipus *Quaternion*[5] que emmagatzemarà la rotació de l'articulació de l'extremitat.
- **parentJointType**. Objecte del tipus *nuitrack.JointType* que emmagatzemarà l'articulació pare reconeguda també per Nuitrack.
- **parentBone**. Objecte del tipus *Transform* que emmagatzemarà l'articulació pare de les extremitats en l'escena.
- **baseDistanceToParent**. Objecte del tipus *float* que emmagatzemarà la distància entre l'objecte bone i el parentBone.

La implementació de tots els atributs explicats anteriorment, ha quedat d'aquesta manera en l'*script ModelJoint.cs* de Unity:

```
[System.Serializable]
public class ModelJoint
{
    public Transform bone;
    public nuitrack.JointType jointType;
    [HideInInspector] public Quaternion baseRotOffset;

    public nuitrack.JointType parentJointType;
    [HideInInspector] public Transform parentBone;
    [HideInInspector] public float baseDistanceToParent;
}
```

LISTING 4.1: Contingut del fitxer ModelJoint.cs

Posteriorment, quedarà crear un altre *script* on s'adjuntarà a l'objecte avatar en l'escena d'Unity. Aquest *script* s'anomenarà *MoveAvatar.cs* i serà l'encarregat de recollir les dades que li proporcioni Nuitrack i realitzarà el corresponent càlcul per poder moure l'avatar dins de l'escena.

L'arxiu *MoveAvatar.cs* comptarà amb un objecte de la classe creada anteriorment (vegeu codi 4.1), on s'emmagatzemarà tota la informació sobre l'articulació la qual farà referència en l'escena d'Unity i la seva articulació pare, si en te. Les articulacions que es seleccionin en l'escena Unity pel seguiment de l'esquelet, seran guardades

[3]Transforms. (2017, 1 d'agost). *Unity | Documentation*. Data de consulta: 17:57, gener 24, 2019 des de https://docs.unity3d.com/es/current/Manual/Transforms.html.

[4]Nuitrack Skeleton Tracker. (2018, 27 de desembre). *Unity | Documentation*. Data de consulta: 18:11, gener 24, 2019 des de http://download.3divi.com/Nuitrack/JointType.

[5]Rotación y Orientación en Unity. (2017, 1 d'agost). *Unity | Documentation*. Data de consulta: 18:03, gener 24, 2019 des de https://docs.unity3d.com/es/current/Manual/QuaternionAndEulerRotationsInUnity.html.

en un diccionari de dades i s'aniran actualitzant en funció del *feedback* rebut en cada articulació. En cada *frame* es comprovarà les articulacions que hagin canviat de posició i rotació, posteriorment es produirà els canvis al diccionari on aquests es veuran afectats directament en l'avatar (vegeu apèndix A.1).

L'agregació de l'*script* anterior en l'avatar, provocarà nous canvis en la configuració de l'avatar. Això implicarà l'aparició de nous camps, que s'hauran d'omplir obligatòriament, ja que serà la informació necessària per produir moviment en ell i que a més, es guardarà en el diccionari creat anteriorment (vegeu figura 4.8). En el primer camp *Size* s'especificarà el nombre d'articulacions que es mouran de l'avatar i amb això també se li donarà mida al diccionari que es crearà posteriorment en temps d'execució. Com a conseqüència, es desplegaran tants elements com mida tingui el diccionari, és a dir, si es posa mida 6, s'afegiran 6 elements (Element 0 - 5). Per cada element apareixerà 3 camps que faran referència als atributs explicats en el codi 4.1, únicament aquells que no tinguin l'atribut [HideInInspector]. En aquest cas únicament es posaran 6 elements, 3 per cada extremitat que seran les espatlles, els colzes i les mans.

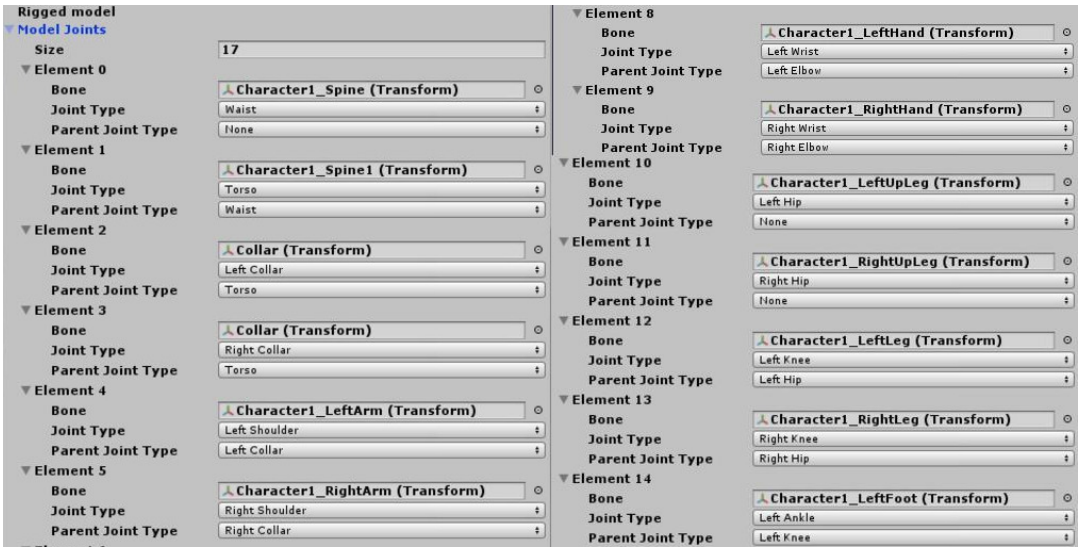



FIGURA 4.8: Exemple de l'agregació de la relació entre objectes de l'escena i les articulacions detectades per Nuitrack

Font: Creació pròpia

Per acabar, únicament quedarà provar que l'avatar es mou seguint els moviments produïts per l'usuari. Una vegada implementats els passos de les anteriors seccions, l'avatar es mourà aproximadament com es mostra en la següent figura 4.9 quan es produeixin moviments amb les extremitats superiors. L'avatar reaccionarà tant si s'aixeca com si s'abaixa el braç, tan si s'aproxima a la càmera com si no, etc. En la creació del projecte, a causa de la introducció del 3D en el projecte, també tindrà en compte la profunditat (eix $z$). Per mitjà dels passos anteriors, es com Nuitrack utilitza la tècnica del seguiment de l'esquelet en Unity 3D.

FIGURA 4.9: Resultat obtingut amb l'agregació de l'*script* (GIF)
Font: Creació pròpia

### 4.1.4 | Dificultats trobades

La conseqüència de la finalització de la iteració, ve donada pels diversos problemes respecte al resultat obtingut en l'avatar i els seus moviments, tal com es pot observar en l'anterior figura 4.9. El primer i un dels problemes minoritaris, fa referència al desplaçament erroni d'algunes articulacions en posicions de l'escena que no corresponen per l'articulació determinada. A més, el problema principal de l'abandonament de la iteració, és a causa de la no assignació de totes les articulacions que componen l'avatar, deixant aquelles de banda que no són menyspreables i això provoca que aquelles que estan en contacte, al produir-se moviment, les textures s'estirin d'una forma poc ortodoxa, fet que això comporta problemes en l'experiència de l'usuari.

Un altre problema menor, és que ocasionalment no detecta amb exactitud algunes posicions en el seguiment d'esquelet, això pot venir provocat a causa del *middleware*, ja que sí que ofereix compatibilitat en molts dispositius però no es creu que funcioni de la mateixa forma amb tots, ni tampoc de la manera més òptima. Per exemple, amb la càmera TVico funciona millor, ja que és el dispositiu que inclou per defecte.

En conclusió, donades les diverses causes anteriors, s'ha decidit començar una nova iteració i per tant, tornar a desenvolupar l'escena de nou on s'intentarà abordar els diferents problemes trobats des d'un altre camí que sigui capaç de solucionar els problemes trobats en aquesta primera iteració.

## 4.2 | Segona iteració

En aquesta segona iteració es partirà de tot el desenvolupament que s'ha dut a terme prèviament en l'anterior iteració i s'intentarà abordar els problemes que han sorgit en la secció 4.1.4 des d'un altre camí. Tanmateix, s'afegiran nous objectius a més dels que hi havia en l'anterior iteració per seguir amb el desenvolupament previst del projecte.

### 4.2.1 | Anàlisis

Respecte els problemes trobats en l'anterior iteració, s'ha decidit prescindir de l'avatar, encara que això no suposarà cap modificació en els objectius relacionats tant d'aquesta iteració, com de l'anterior. S'elegirà una alternativa més útil per el projecte en qüestió i que serà explicada en la següent secció 4.2.2. Així doncs, es procedirà a seguir amb el desenvolupament previst del projecte per aquesta iteració.

Així mateix, els requeriments que s'assignaran en aquesta iteració, a més del que ja hi havia prèviament en la iteració anterior (vegeu secció 4.1.1), són els següents:

- Afegir un ressaltat o *highlight* a la figura al produir-se la col·lisió
- Moure la figura a partir de les col·lisions produïdes per l'avatar

Aquests objectius fan referència a la *issue* anomenada "Main scene: Iteration 2" de la secció 3.2. Per tant, l'objectiu principal d'aquesta iteració serà completar els objectius de la iteració anterior que no s'han dut a terme i completar els de la iteració actual de manera satisfactòria.

### 4.2.2 | Disseny

La primera decisió de disseny que s'implementarà està relacionada amb la modificació del disseny de l'avatar. Tenint en compte els problemes que s'han trobat en l'anterior iteració respecte a l'avatar, s'ha optat per canviar tot el disseny i únicament implementar dues extremitats superiors, ja que són amb les que es produeixen la interacció en l'escena. S'ha decidit prescindir d'aquelles parts menyspreables que no s'utilitzaven per evitar de crear els problemes trobats en les textures en la primera iteració (vegeu secció 4.1.4).

La segona i última decisió de disseny fa referència a l'entorn, es canviarà la temàtica cap a un context més ortodox i que sigui perfectament combinable amb l'avatar. Encara que no s'havia invertit gens en el desenvolupament de l'entorn en la primera iteració, s'ha decidit canviar la temàtica de l'entorn perquè combini amb l'avatar i la figura a manipular.

Per acabar, les físiques que s'havien d'implementar en la primera iteració, ja que encara no s'ha dut a terme, no presentaran nous canvis i se seguirà amb la proposta d'implementació introduïda en la primera iteració, on també forma part dels objectius d'aquesta.

FIGURA 4.10: Disseny aproximat de les extremitats superiors
Font: https://forum.unity.com

### 4.2.3 | Desenvolupament

Respecte al desenvolupament, com s'ha esmentat anteriorment, es partirà des de la creació de tots dels scripts que han fet possible el moviment de l'avatar i s'han dut a terme en la primera iteració. Únicament, es disposarà de l'escena amb els scripts creats i els paquets que s'han importat prèviament de l'Asset Store d'Unity.

#### Modificació de l'avatar

Com s'ha esmentat anteriorment, es canviarà l'avatar que hi havia prèviament en l'escena, per unes extremitats que simularan les d'un ésser humà i que realitzaran la mateixa funció. Aquestes extremitats s'importaran prèviament d'una pàgina externa[24]. Aquest paquet conté un arxiu Filmbox (FBX) i conté les extremitats de l'avatar en forma digital, únicament serà necessari importar-ho a Unity per agregar-les en l'escena.

Una vegada s'ha agregat les dues extremitats en l'escena, caldrà tornar a enllaçar cada articulació amb els diferents elements del diccionari que enllaçarà els *assets* de l'escena amb la posició que s'obtindrà del sensor (vegeu secció 4.1.3). Posteriorment, les extremitats seran capaces d'imitar els moviments que realitzi l'usuari davant del sensor.

#### Creació de l'entorn

Amb la necessitat d'adquirir una nova temàtica respecte a l'entorn, s'ha decidit enfocar l'entorn en una oficina. El *prefab* s'obtindrà des d'una pàgina de tercers on facilita el paquet d'objectes[25].

La idea inicial és crear una habitació amb diversos elements, els quals es destacarà la figura a modelar (que serà un cub) i una taula, on s'interactuarà amb la figura que es trobarà sobre d'ella. Tenint en compte que les extremitats ja estaran dins de l'escena, tant la figura com la taula s'hauran de posar en una àrea acció propera a l'avatar perquè aquest pugui ser capaç de tocar la figura i la taula amb les mans sense la necessitat de què l'usuari canvi de posició davant del sensor.

Com s'observa en la següent figura 4.11, l'entorn es generarà a partir dels paquets que han estat importats prèviament i tot seguit s'ambientarà en un entorn d'una oficina. L'espai de l'oficina es generarà a partir d'un *prefab* que s'ubica dins del paquet *Office Mega Kit*. No s'inclouran més elements d'entorn per ara, ja que implica també més temps a l'hora de construir i executar el projecte, per tant, no valdrà la pena gastar molts recursos en l'entorn, ja que és una cosa secundària.

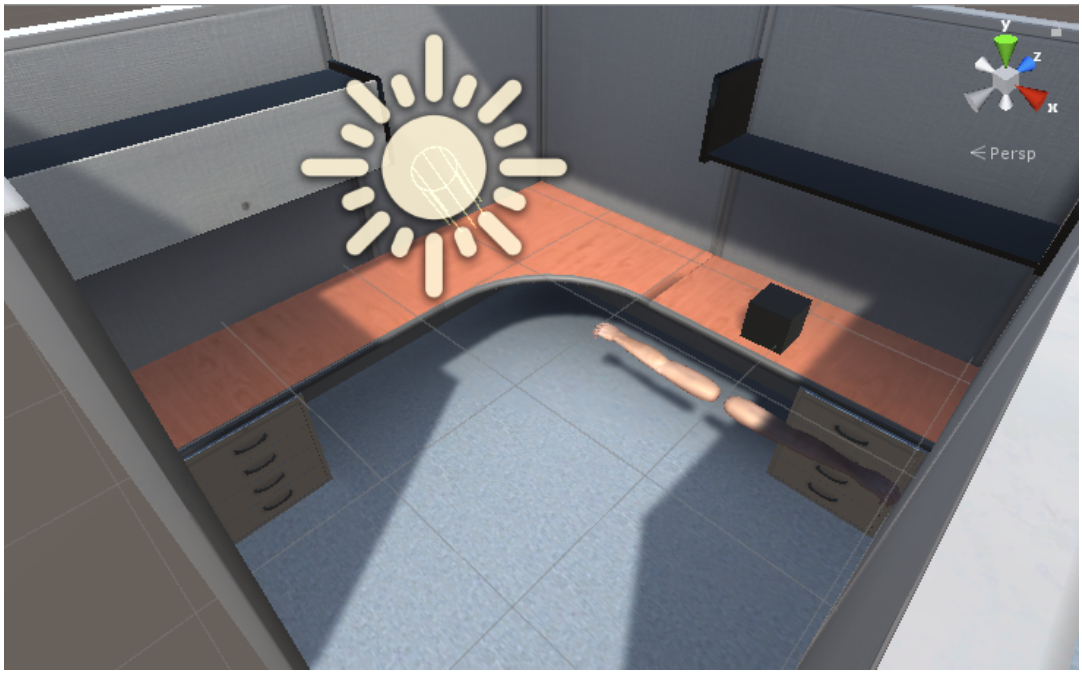



FIGURA 4.11: Entorn generat basat en la temàtica d'una oficina
Font: Creació pròpia

### Càmera en primera persona

Un dels punts importants del prototipus és veure com es visualitzarà l'escena, és a dir, com ho veurà l'usuari que interactua davant del sensor amb el prototip. Cal tenir present que en qualsevol creació d'una nova escena en un projecte d'Unity 3D, apareixen dos objectes per defecte que són:

- ***Main camera***. És aquell dispositiu que captura i ensenya l'escena la qual es troba la càmera a l'usuari que fa ús de l'aplicació[6].
- ***Directional Light***. És la llum artificial que proporciona Unity en una escena. Amb aquest objecte es pot donar color a l'escena i generar les respectives ombres que genera cada objecte en una escena. Es podria dir que és una font de llum dins d'una escena[7].

[6]Cámara. (2018, 3 de juliol). *Unity | Documentation*. Data de consulta: 20:28, gener 29, 2019 des de https://docs.unity3d.com/es/current/Manual/class-Camera.html.

[7]Lights (luces). (2018, 3 de juliol). *Unity | Documentation*. Data de consulta: 20:30, gener 29, 2019 des de https://docs.unity3d.com/es/current/Manual/Lights.html.

Respecte a la llum direccional, no es farà molta èmfasi, ja que s'utilitzarà la llum que proporciona el *prefab* en alguns objectes de l'entorn de l'escena i que s'ha posat prèviament. En canvi, sí que es canviarà alguns aspectes sobretot de la visualització de la càmera, ja que ara mateix el mode de visualització és en tercera persona, és a dir, la càmera de l'usuari està situada davant de l'avatar. S'implantarà el mode en primera persona perquè es vol aconseguir que la interacció amb el prototip sigui el més realista possible apropant a l'usuari a una tecnologia el més immersiva possible. Existeixen dues formes de dur-ho a terme que són:

1. **Adherint el component càmera en un objecte**. S'agregarà un component càmera en un objecte *head* que es crearà prèviament i que realitzarà els mateixos moviments que l'usuari faci amb el cap davant del sensor. És a dir, s'afegirà una càmera en un objecte que se situarà per damunt de les dues extremitats, intentant simular com si la captura de la càmera fos els ulls de l'usuari.

   El problema que presenta aquest mètode és la sensibilitat al moviment que provoca i implicaria introduir una nova articulació, en aquest cas el cap. En aquest cas, on miri l'usuari, la càmera el seguirà provocant un moviment continu en la imatge on a vegades pot resultar molest. Aquest moviment no està molt ben detectat per part del SDK de Nuitrack, ja que no és molt sensible a la rotació del cap.

2. **Situant una càmera fixa en la posició del cap**. La idea d'aquest mètode, és situar la càmera principal que ve en l'escena d'Unity a la posició que ocuparia el cap d'una persona tenint com a referència les dues extremitats. Amb la implementació d'aquest mètode, se solucionarà els problemes presentats que sorgien en l'anterior implementació. En aquest cas, la càmera serà fixa i no dependrà de cap objecte que estigui adherida i que provoqui moviments. L'únic inconvenient que podria presentar és que l'objecte a modelar surti fora del camp de visió de la càmera.

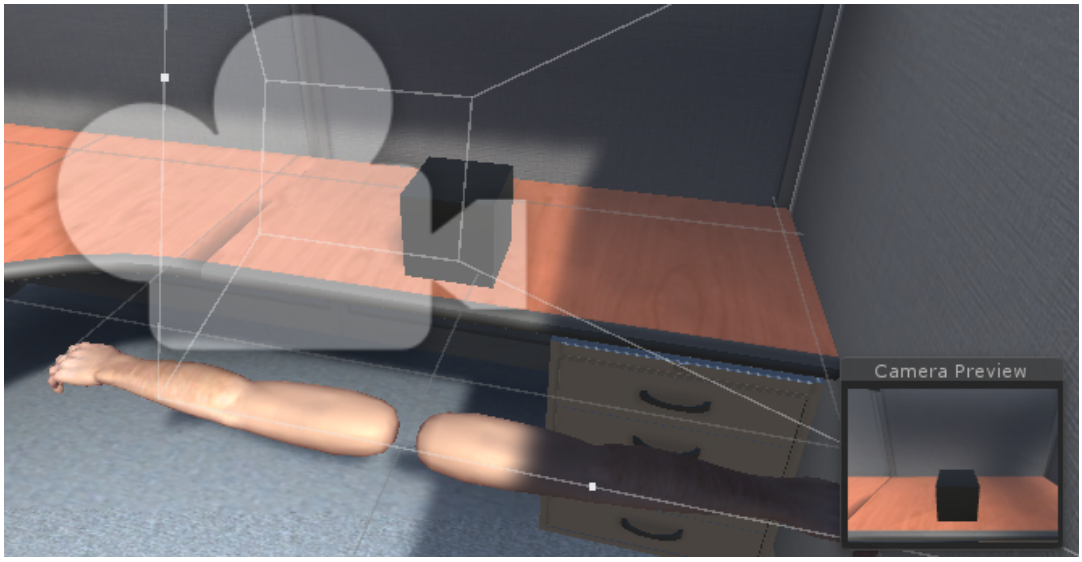



FIGURA 4.12: Ubicació de la càmera principal sobre les extremitats
Font: Creació pròpia

La forma escollida en la qual es durà a terme en el projecte serà la segona, ja que a simple vista presenta menys problemes que no pas la primera. Tanmateix, una vegada descartada la implementació de l'avatar en la primera iteració i per tant, l'eliminació de totes les articulacions exceptuant les dues extremitats superiors, impossibilita la primera solució, ja que no es disposa de l'objecte que faci la funció del cap.

Per ubicar correctament la càmera, es tindrà en compte la posició dels eixos $x$ i $z$ de les extremitats. Aquesta posició serà una mitja de les posicions de les dues articulacions arrels (les espatlles) i posteriorment s'afegirà valor a l'eix $y$ per donar altura a la càmera. Una vegada situada més o menys quedarà com apareix en la següent figura 4.12.

Com a desavantatge d'escollir la segona solució, és la pèrdua de realisme en l'experiència de l'usuari, ja que no depèn del moviment del cap i únicament és una càmera fixa en un punt. Tanmateix, la possible aparició de les dues extremitats en l'escena a causa del moviment de l'usuari en l'eix $z$ (la profunditat de l'usuari en el sensor), provoca l'aparició íntegra de les extremitats en el camp de visió de la càmera de l'escena d'Unity i provoca la pèrdua de realisme en la interacció (vegeu figura 4.13).

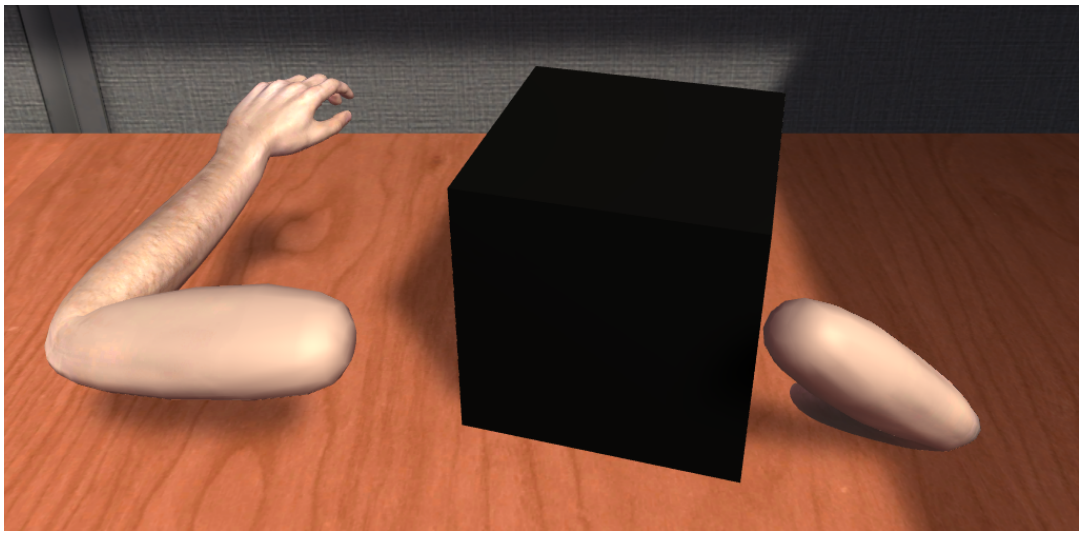


FIGURA 4.13: Problemes generats a causa de la profunditat (eix $z$)
Font: Creació pròpia

**Agregació de físiques: *rigidbody***

Una vegada s'ha arribat en aquest punt, ja és possible moure les extremitats amb els moviments produïts davant del sensor però, si es toca l'objecte, per què traspassa? Doncs bé, per evitar aquesta mena de situacions, serà necessari implementar un component a l'objecte de l'escena, que li proporcionarà un cos físic (*rigidbody*).

Aquest component implica, que serà sensible a la gravetat o altres forces que puguin provenir d'altres objectes, com per exemple les mans de l'avatar. S'implementarà el *rigidbody* en aquells objectes que estaran afectats per la gravetat o alguna força del motor de joc (característica *UseGravity*), o bé que es variaran la posició des d'algun *script* o pel component *Transform* i no pel mateix motor de físiques d'Unity 3D (característica *IsKinematic*)[8].

Principalment s'afegirà el component en el cub i es marcarà la casella de *UseGravity* perquè sigui susceptible a la força de la gravetat. En la resta de l'entorn no serà necessari aplicar les forces de la gravetat perquè les col·lisions que hi pugui haver seran menyspreables en aquest prototip. Per altra banda, s'afegirà el component i es marcarà amb l'opció *IsKinematic* a totes les articulacions que formen part de les extremitats, des de la part superior del braç fins a la punta dels dits. Aquests no estaran controlats per cap força del motor de joc, sinó que seran moguts pels valors que s'obtingui del *script MoveAvatar.cs* (vegeu apèndix A).

Posteriorment, caldrà implementar un script perquè el component *Rigidbody* segueixi l'objecte al qual està associat, tant la posició, com en rotació. Per fer-ho possible s'implementarà un script amb el següent contingut:

```
public class RigidBodyFollower : MonoBehaviour
{
    [SerializeField] Transform target;
    Rigidbody rigidbody;

    // Start is called before the first frame update
    void Start()
    {
        rigidbody = GetComponent();
    }

    void FixedUpdate()
    {
        rigidbody.MovePosition(target.position);
        rigidbody.MoveRotation(target.rotation);
    }
}
```

LISTING 4.2: Script pel seguiment del *Rigidbody* a l'objecte associat

### Agregació de malles de col·lisió: *colliders*

Un element necessari i indispensable perquè l'avatar sigui capaç de provocar *feedback* en l'objecte que manipularà, a part del *rigidbody*, serà una malla de col·lisió. Aquest component s'utilitzarà per a aquells propòsits de col·lisió entre els objectes de l'escena[9]. En un primer moment es pot deduir que els dos objectes que seran

[8] Rigidbody. (2017, 17 de juliol). *Unity | Documentation*. Data de consulta: 18:26, gener 29, 2019 des de https://docs.unity3d.com/es/current/Manual/class-Rigidbody.html.

[9] Colliders. (2017, 1 de juliol). *Unity | Documentation*. Data de consulta: 18:31, gener 29, 2019 des de https://docs.unity3d.com/es/current/Manual/CollidersOverview.html.

afectats per una col·lisió, seran les dues extremitats i el cub, però també serà necessari implementar el *collider* a la taula que s'encarrega d'aguantar la figura a modelar perquè de no ser així, la figura que estarà afectada per la gravetat, no col·lisionarà amb cap element de l'entorn i caurà al buit. Per altra banda, en l'agregació del cub en l'escena, aquest ja venia amb un *Box Collider* per defecte, ja que és un objecte que ja venia inclòs en la distribució d'Unity, així que no serà necessari implementar un *collider* per l'objecte a manipular.

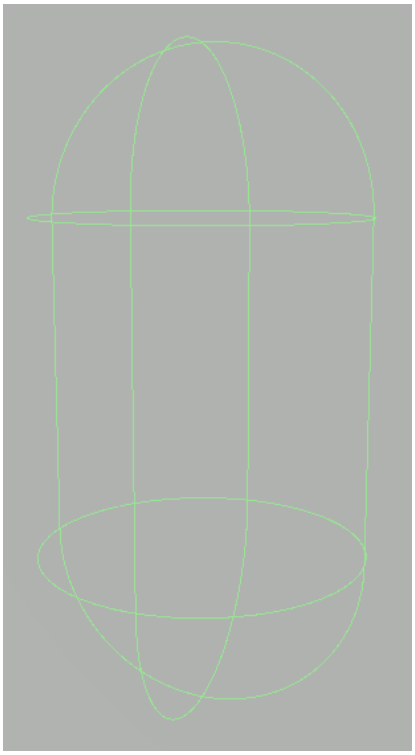


FIGURA 4.14: Exemple de *Capsule Collider* en l'escena d'Unity
Font: https://ekulabo.com

Respecte a les extremitats, no és un objecte que vingui per defecte en Unity i per tant, no disposa d'un *collider* tan bàsic, com per exemple seria un cub o una esfera. Una de les possibilitats que existeix per incorporar un *collider* en un objecte importat, seria fent-ho per mitjà d'un *Mesh collider* que no és res més que una malla que s'adapta a la forma de l'objecte en la qual està associada. La idea és passar-li l'objecte 3D de cada extremitat al *Mesh Collider* i ell mateix s'encarregarà d'extreure el *collider* en funció dels vèrtexs que conté l'objecte. Aquesta forma pot ser perjudicial en funció del nombre de vèrtexs que contingui l'objecte, ja que per calcular una col·lisió, si l'objecte conté un nombre molt elevat de vèrtexs, pot provocar un càlcul massiu i posteriorment la sobrecàrrega en el motor de joc. En la següent figura 4.15 es pot observar la implementació del *Mesh Collider* on genera prop de 3000 vèrtexs.

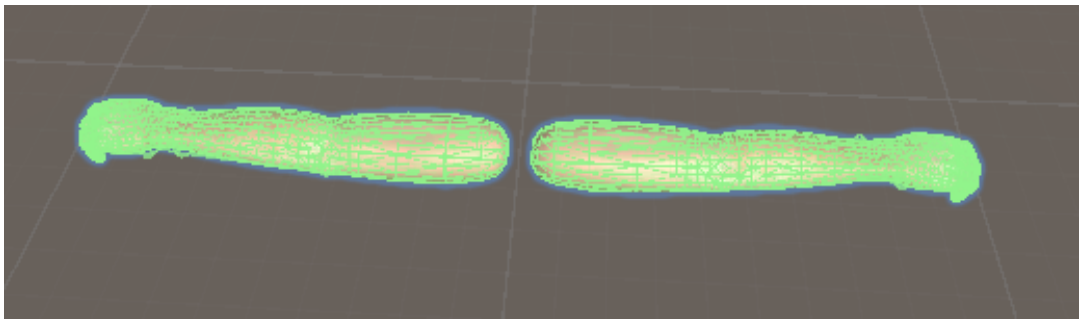


FIGURA 4.15: Implementació del *Mesh Collider* en les extremitats
Font: Creació pròpia

En aquest projecte s'implementarà els *colliders* en les extremitats de forma més simple i no provocarà cap sobrecàrrega en el motor de joc. S'utilitzaran *Capsule Colliders* en les diferents parts de les extremitats, com les que apareixen en la figura 4.14.

Aquestes estan compostes per dues semiesferes i un cilindre. S'ha escollit aquest tipus de figura perquè és la que més s'encaixa amb el disseny de les extremitats i no conté un nombre molt elevat de vèrtexs. En cada extremitat s'afegiran un màxim de 3 *Capsule Colliders* que formaran part de la part superior del braç, l'avantbraç i la mà.

Com a resultat s'obtindrà un disseny com el que apareixerà en la següent figura 4.16 la qual s'ha adoptat una mida diferent de cadascuna de les càpsules perquè encaixin amb l'articulació que li correspon. És obvi que no encaixaran al 100%, però únicament s'introduiran aquest tipus de figures perquè simularan un comportament més real a la possible col·lisió amb el cub a modelar i menys sobrecàrrega al motor de joc[1].

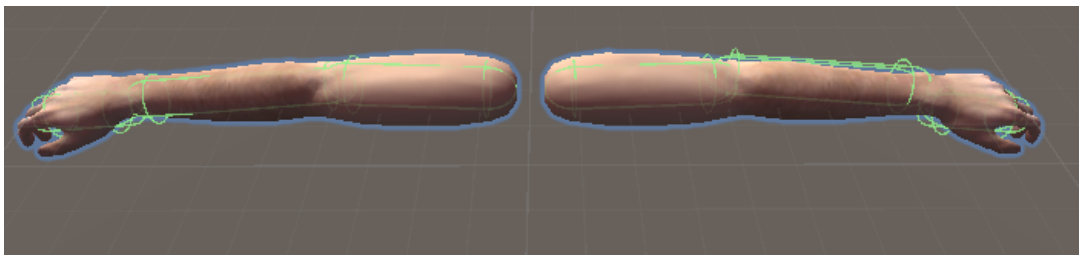


FIGURA 4.16: Implementació dels *colliders* en les extremitats
Font: Creació pròpia

**Agregació d'un ressaltat en la figura**

Amb l'agregació d'un ressaltat o *highlight* en la figura a modelar, es vol provocar *feedback* per part del sistema quan l'usuari col·lisiona directament amb l'objecte. La idea és que aquest ressaltat que emeti la figura, vagi en funció de si l'usuari està o no a prop de la figura, o si està en contacte.

Per realitzar aquestes característiques, s'ha optat per un paquet de *assets* aconseguit des d'una pàgina de tercers, on és possible dur a terme els objectius esmentats anteriorment[26]. Posteriorment, serà importat de manera externa tal com s'ha explicat anteriorment. El paquet *Highlighting System* està format per un script encarregat de dur a terme tota la funcionalitat sobre el remarcat en l'objecte el qual estarà associat i un conjunt d'ombrejats (*shaders*) que seran els arxius que efectuaran la il·luminació òptima en cada moment a partir de càlculs matemàtics[10].

Per l'agregació del remarcat al cub, únicament caldrà agregar l'únic script del qual disposa el paquet de *assets* en qüestió. En aquest cas, s'aprofitarà la implementació tant de les malles de col·lisió, com de les físiques per activar el ressaltat de la figura. La idea és que quan la malla de col·lisió estigui entrant en contacte amb la figura, el ressaltat s'activi de forma automàtica. Per altra banda, també s'implementaran funcions que seran disparadors d'esdeveniments, conseqüentment quan entri l'avatar en l'àrea de contacte amb la figura, s'executarà la següent funció:

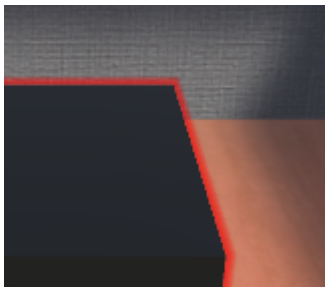
[10]Materiales, Shaders & Texturas. (2017, 26 d'octubre). *Unity | Documentation*. Data de consulta: 10:18, abril 10, 2019 des de `https://docs.unity3d.com/es/current/Manual/Shaders.html`.

```
private void OnCollisionEnter(Collision collision)
{
    if (collision != null)
    {
        if (!list.Contains(collision.gameObject.tag) && collision.
            gameObject.tag != "Table") {
            count++;
            list.Add(collision.gameObject.tag);
        }
        if (count >;;= 0)
            cube.GetComponent().lightOn();
    }
}
```

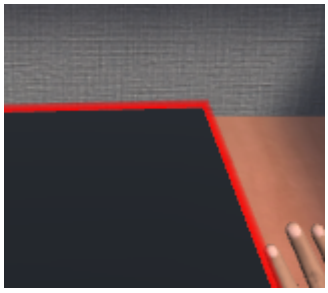
LISTING 4.3: Funció OnCollisionEnter del fitxer Highlight.cs

En el Codi 4.3 s'utilitza simplement per marcar el ressaltat a la figura o en el cas contrari (funció *OnCollisionExit*), per desmarcar-lo. Per dur-ho a terme, es tenen una sèrie de comprovacions respecte al ressaltat com per exemple, el valor *null* o tenir un control de quina és l'extremitat que entra en contacte per tal de no crear més instàncies de les necessàries de *shaderGlow*.

La intensitat del ressaltat vindrà donada per l'última comprovació de la funció anterior, ja que si posem les dues extremitats en l'àrea de contacte amb la figura, la llum serà més intensa gràcies al nombre de crides que es faci del mètode *lightOn()* i per tant, el ressaltat serà més intens (vegeu figura 4.17b). En canvi, si només s'entra en contacte amb una extremitat o es produeix una aproximació cap a la figura, la llum no serà tan intens respecte el cas anterior (vegeu Figura 4.17a).



(A) Ressaltat amb una intensitat baixa



(B) Ressaltat amb una intensitat alta

FIGURA 4.17: Tipus de ressaltats en la figura a modelar

**Moviment de la figura mitjançant l'avatar**

Una vegada s'han dut a terme totes les implementacions anteriors, produir moviments a la figura que es vol manipular, no requereix modificar ni programar res. Des de la implementació de les malles de col·lisió (*colliders*) i el cos físic als objectes (*rigidbody*), ja és possible produir canvis de posició en la figura. Una vegada s'han posat els elements necessaris per provocar la col·lisió és el mateix motor de joc el que s'encarrega tant de calcular-la, com de dur-la a terme.

En Unity quan detecta la col·lisió de l'avatar sobre l'objecte, calcula internament el desplaçament de la figura en funció de la velocitat, direcció i d'altres paràmetres, de caràcter rellevant, en què s'ha produït l'impacte. Amb tot això, es calcula posteriorment el desplaçament que es donarà a la figura i es mostrarà en l'escena tridimensional.

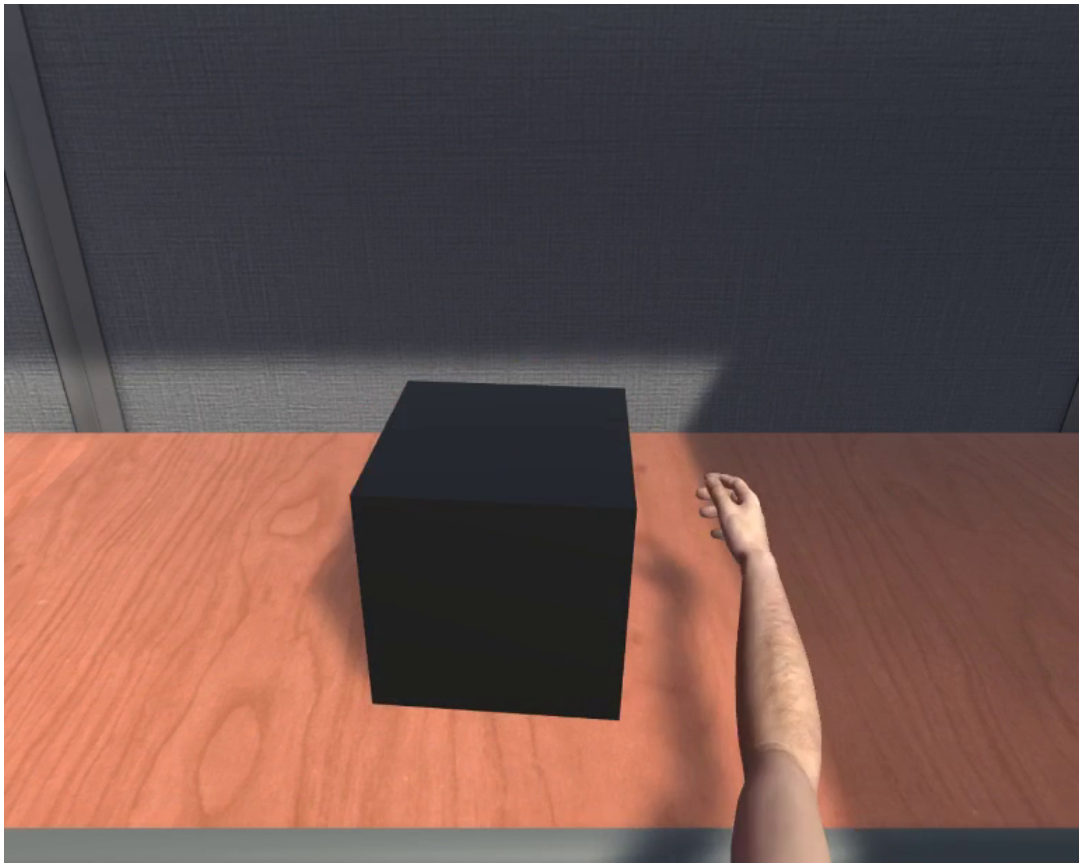


FIGURA 4.18: Resultat obtingut amb el moviment de la figura (GIF)
Font: Creació pròpia

Amb l'agregació del ressaltat, s'augmenta el *feedback* que dóna el sistema a l'usuari que l'utilitza, ja que abans de produir-se el moviment de la figura, l'avatar entra en contacte amb aquesta; fet que s'executa el disparador del ressaltat sobre la figura just abans del moviment.

### 4.2.4 | Dificultats trobades

En aquesta segona iteració, s'ha finalitzat gràcies al compliment de tots els objectius que s'han marcat prèviament en la fase de disseny (vegeu secció 4.2.2), abans d'inicialitzar el desenvolupament d'aquesta. S'han dut a terme tant els objectius de la primera iteració, com els de la iteració actual.

Els problemes que s'han anat trobant al llarg del desenvolupament, corresponen a aspectes d'interacció i en part d'experiència d'usuari, com és el cas de la figura 4.13, on s'observava el problema amb les extremitats de l'avatar que entren completament dins de l'àrea de visualització de l'escena.

Per altra banda, també hi va haver dificultats en implementar les malles de col·lisió. La idea en un primer moment era fer-ho mitjançant un *Mesh Collider* per tenir en tot moment una àrea de col·lisió al voltant de la textura de l'avatar i donar-li el màxim realisme possible, però, el problema de tot això, és que no reaccionava als canvis de forma i posició de les extremitats, sempre es quedava de la forma en la qual apareixia en la figura 4.15. Així que es va prescindir de realitzar-ho en una *Mesh collider* i s'ha realitzat en *Capsule Colliders* (vegeu figura 4.16).


FIGURA 4.19: Videojoc de futbol en realitat virtual
Font: https://www.españavirtual.org

Com a petits canvis que es poden dur a terme en la següent iteració, seria una altra modificació de l'avatar en les dues extremitats superiors, ja que els problemes que podia presentar, és que les extremitats puguin tapar gran part del camp de visió de l'usuari a l'hora de manipular la figura a causa dels problemes que sorgien de proximitat o allunyament de l'usuari (l'eix $z$ en l'escena). Una idea o solució que es podria dur a terme és implementar únicament les mans. Pràcticament avui en dia qualsevol videojoc que incorpori realitat virtual, disposa d'un dispositiu d'entrada en cada mà de l'usuari, que no deixa de ser res més que un comandament per manipular els objectes dins de l'entorn virtual. Això es transmet en el videojoc com unes mans on l'avatar interacciona amb els elements de l'escena. En l'exemple de la Figura 4.19 es pot observar com els guants del porter són les mans que estan controlades per mitjà dels dispositius d'entrada que té l'usuari. La idea en aquest projecte seria de fer-ho de forma no invasiva, a diferència dels jocs en realitat virtual.

## 4.3 | Tercera iteració

En aquesta tercera i probablement, la iteració definitiva es partirà del treball realitzat en l'anterior iteració i s'intentarà solucionar els problemes que s'han trobat en la secció 4.1.4. Igual que en les anteriors iteracions, s'afegiran nous objectius que formaran la conclusió del desenvolupament d'aquest prototip.

### 4.3.1 | Anàlisis

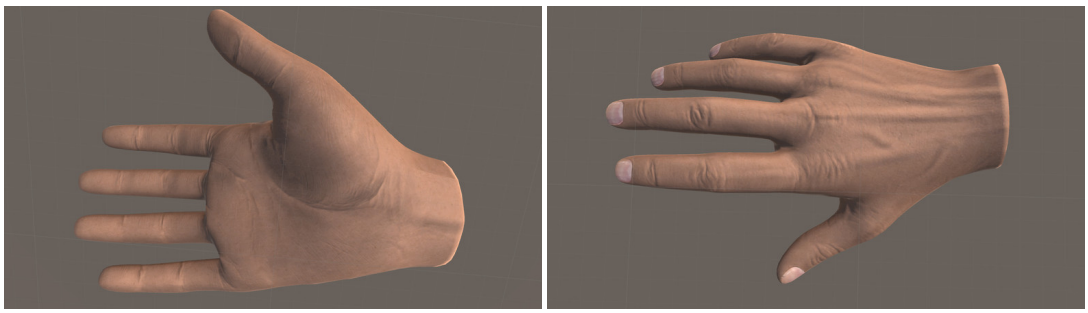
En aquesta iteració els últims requisits per concloure el projecte de forma satisfactòria, estaran relacionats amb la manipulació de la figura en l'escena i l'objectiu principal del projecte. Tanmateix, també es modificarà la figura de l'avatar deixant tan sols unes mans com les que es veuran en la secció 4.3.2. Per tant, els objectius de la tercera iteració són:

- Implementació de la tècnica de reconeixement de gestos en l'avatar
- Agafar i desplaçar l'objecte en que es produeix la interacció
- Deformar/modelar l'objecte en que es produeix la interacció

Aquests objectius fan referència a la *issue* anomenada "Main scene: Iteration 3" de la secció 3.2. Per tant, l'objectiu principal d'aquesta iteració serà completar tots els objectius del projecte i els petits canvis o modificacions que puguin existir respecte a les iteracions anteriors.

### 4.3.2 | Disseny

Respecte als canvis proposats en l'anterior iteració, es durà a terme un canvi en el disseny de l'avatar en el prototip. Aquest consistirà en un canvi notori en les extremitats on bàsicament, es prescindirà dels braços quedant únicament les mans on es realitzarà el 100% de la interacció (vegeu figura 4.20).



(A) Palmell de la mà (B) Dors de la mà

FIGURA 4.20: Exemple de mà utilitzada en la realitat virtual

Aquest canvi implica que s'hagi de tornar a agregar un altre cop les malles de col·lisió i també el cos físic per produir les col·lisions en els altres objectes de l'escena. Com que s'ha explicat anteriorment el procediment per dur-ho a terme, no es farà èmfasi en aquesta iteració a excepció que es produeixin canvis molt notoris.

Per la resta d'elements de l'escena, com la figura a manipular o l'entorn, no es durà a terme cap modificació respecte al seu disseny. El disseny que s'ha dut a terme en la iteració 2, serà suficient per seguir amb el desenvolupament previst. La idea en un futur, és canviar la temàtica de l'entorn a *low poly*. Això significa, utilitzar objectes en l'escena que tinguin un baix cost de computació tan per afegir-los com de generar-los, ja que són a simple vista, objectes molt simples.

### 4.3.3 | Desenvolupament

En el desenvolupament d'aquesta tercera iteració, es partirà de la implementació que s'ha dut a terme en la iteració anterior per acabar de completar els objectius restants del projecte en aquesta, si és possible.

#### Modificació de l'avatar

Com s'ha esmentat en la secció 4.3.2, es canviarà el disseny de l'avatar per unes mans. Aquest disseny serà importat de forma externa a Unity mitjançant l'obtenció del paquet d'*assets* des d'una pàgina de tercers[27].

El paquet anomenat *Hands for VR: Basic* està compost per un conjunt d'animacions, materials, *prefabs*, malles i *scripts*. Hi ha tots els elements necessaris per treballar i agregar correctament les dues mans en l'escena. Una vegada s'ha afegit correctament les mans en l'escena, caldrà tornar a enllaçar cada objecte amb Nuitrack i afegir correctament tant el *rigidbody*, com les malles de col·lisió.

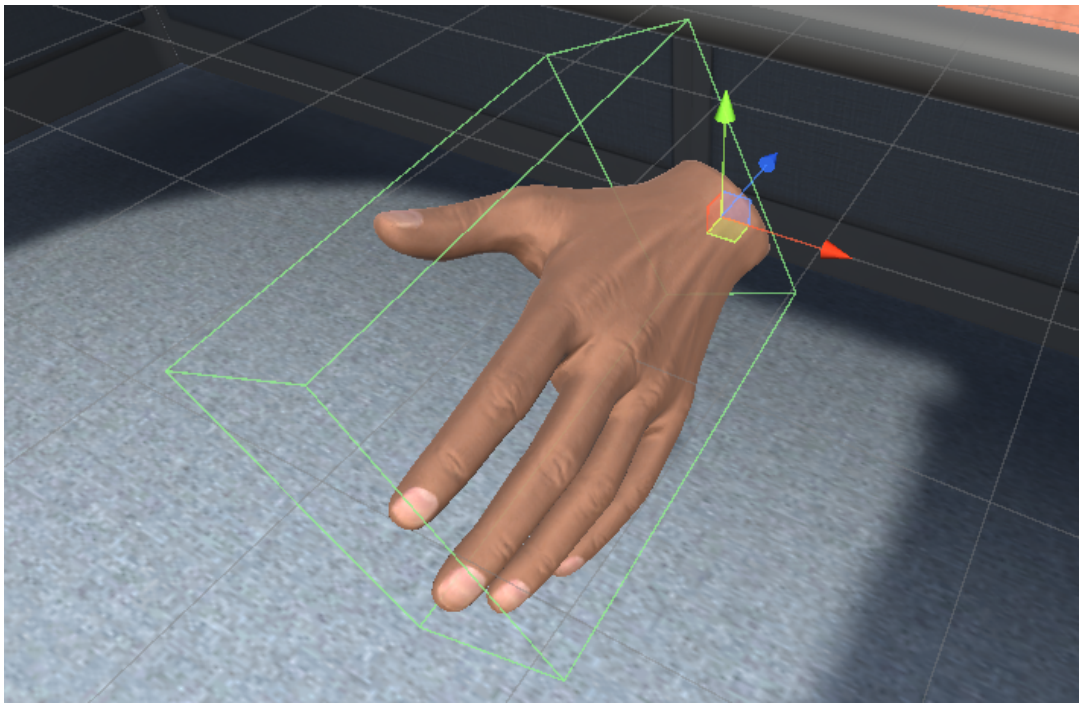


FIGURA 4.21: Implementació del *Box Collider* en les mans de l'avatar
Font: Creació pròpia

Cal esmentar dels canvis que es produiran respecte a les malles de col·lisió en aquest cas i és que, per si sol, un *capsule collider* genera massa àrea de col·lisió i deixa massa espai respecte a la mà. Llavors s'optarà per canviar-lo per un *box collider*, tal com es mostra en l'anterior figura 4.21, on l'espai serà més acurat i augmentarà el realisme a l'hora d'interactuar amb l'objecte.Aquest *Box Collider* anirà complementat amb el script de seguiment de les malles de col·lisió, com en la secció 4.2.3, on seguirà en tot moment al *rigidbody* d'aquella extremitat a qui estigui associada. En un futur, no es descarta que aquest tipus de bloc de col·lisió sigui reemplaçat per un *Mesh collider*, ja que es prioritzarà més que hi hagi una interacció més realista.

### Agregació del reconeixement de gestos i animacions

Com s'ha explicat en la secció 2.2.5, el *middleware* de Nuitrack incorpora el reconeixement de gestos i això es voldrà aprofitar en aquest prototip, per aportar més interacció al sistema.

Per activar el reconeixement de gestos, serà necessari anar a l'objecte arrossegat prèviament a escena anomenat *NuitrackScripts* i marcar l'opció de "Gestures Recognizer Module". Una vegada activat, el *middleware* serà capaç de detectar quan s'obra i es tanca la mà, per exemple.

Aprofitant aquesta nova característica, s'afegirà el gest d'obrir i tancar a les dues mans i s'associaran amb una animació que ve incorporada amb el paquet que s'ha importat prèviament. Cada objecte que forma l'avatar, sigui la mà esquerra o dreta, vénen incorporats en el paquet d'*assets* les animacions per cadascuna d'elles. El contingut d'aquestes animacions es pot observar en la figura 4.22-B.

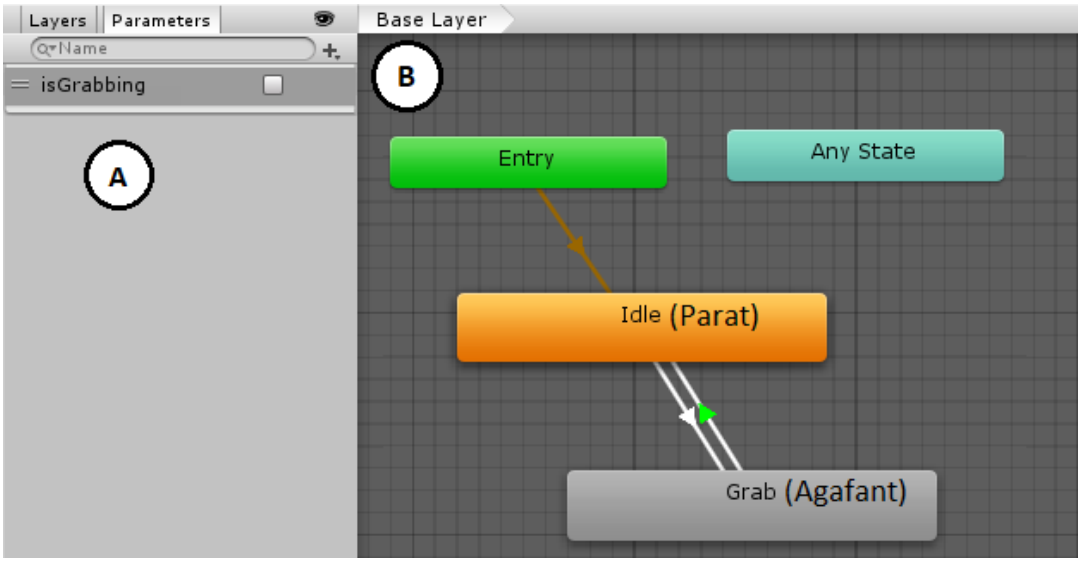



FIGURA 4.22: Contingut de l'objecte d'animació de cada mà inclòs en el paquet d'*assets*
Font: Creació pròpia

Les transicions que es poden dur a terme en aquest avatar com es pot observar són mínimes, obviant el *Entry* que únicament s'executarà a l'inici i no incorpora cap animació en l'objecte, els estats pels quals passarà l'avatar durant l'execució del prototip serà de *Idle* a *Grab* i a l'invers. Perquè es produeixi aquesta transició, serà necessari canviar el valor de la variable booleana associada als paràmetres de l'animació (vegeu figura 4.22-A).

Aprofitant aquesta variable booleana que incorpora el paquet d'*assets*, s'associarà el valor que s'obtindrà de cada mà a l'hora de reproduir un gest, amb la variable booleana anterior. Això provocarà que quan s'obri o es tanqui la mà, es produeixi l'animació corresponent. Per dur-ho a terme serà necessari crear un altre *script* que s'encarregarà de detectar si es produeix o no algun gest en alguna de les dues mans de l'avatar i posteriorment, associar aquest retorn a una transició de les animacions de l'avatar per mitjà de la variable booleana (vegeu apèndix A.2).

Aquesta implementació, serà de gran utilitat en la propera secció, ja que incorporà més realisme en la interacció i es disposarà de més interaccions sobre la figura a manipular.

### Agafar i desplaçar la figura

Una primera idea és agafar la figura quan es produeixi l'animació en qualsevol de les dues extremitats (vegeu secció 4.3.3). Una vegada s'agafa la figura, aquesta seguirà tots els moviments que realitzi la mà de l'avatar en el mateix instant de temps. Aquesta acció finalitzarà quan es deixi anar l'objecte, és a dir, quan s'obri la mà.

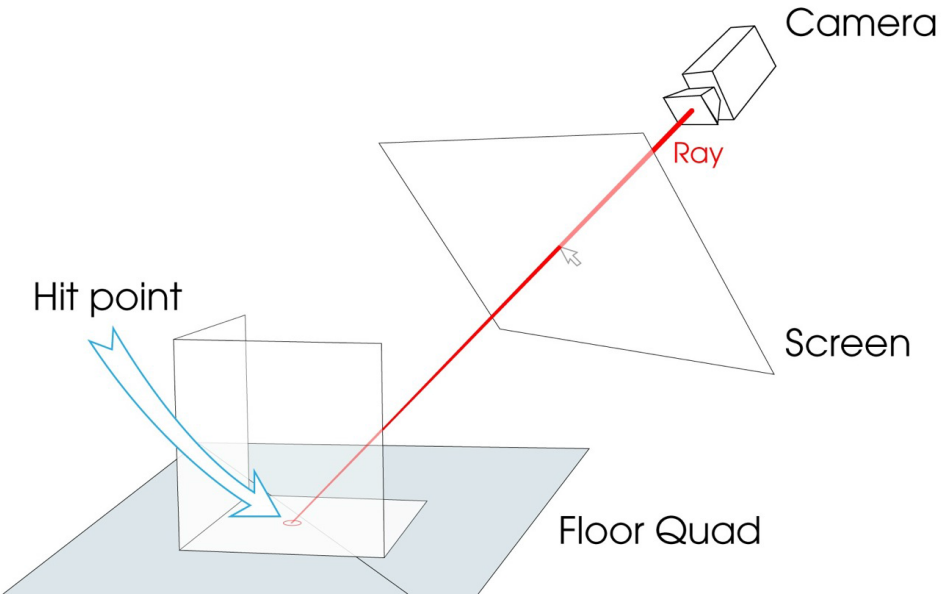



FIGURA 4.23: Exemple visual de la funció del *Raycast*
Font: http://pievisdev.blogspot.com

Un possible camí a tenir en compte i segons la recerca que s'ha dut a terme en aquest àmbit, és començar la interacció amb l'objecte a manipular mitjançant el cursor del sistema. La funcionalitat passa per, quan s'agafa l'objecte amb el botó esquerre i mentre aquest es mantingui premut, es simularà l'acció d'haver agafat l'objecte. Quan es deixi de prémer el botó es realitzarà l'opció de soltar l'objecte[28]. Existeix també, una altra alternativa a la idea de solució anterior sense utilitzar el *Raycast*[11]. La idea es podria dur a terme mitjançant els *colliders* implementats en els diferents objectes de l'escena anteriorment. Però de moment, aquesta possible solució no es tindrà en compte.

Per començar amb la implementació d'aquesta forma s'utilitzaran propietats que proporciona Unity com són els *Raycast* per agafar informació de l'objecte al qual s'-ha produït l'impacte i per tant, en aquest cas, es voldrà agafar. Una vegada capturat aquest objecte, serà necessari que segueixi les coordenades de la mà de l'avatar que l'ha agafat i procurar no provocar contacte entre els dos *GameObjects* (l'objecte a modelar i la mà de l'avatar). L'exemple anterior de la figura 4.23 mostra la funcionalitat del *Raycast* amb el cursor de sistema explicat anteriorment.

Els problemes que presenta la solució amb la implementació del cursor en una mà són varis. El primer d'ells és la limitació de la interacció en una mà, ja que únicament existeix un cursor en la pantalla i aquest únicament pot estar assignat a una extremitat de l'avatar en el mateix instant de temps. Tanmateix, si es realitza un *Raycast* sobre el cursor, l'objecte en el qual anirà l'impacte serà la mateixa mà de l'avatar, per tant, s'haurà de prescindir de fer-ne ús i intentar realitzar el *Raycast* sobre les extremitats de l'avatar, prescindint del cursor.

La idea depèn de capturar tot l'objecte quan es produeixi l'impacte (o *hit*) i posteriorment, es treballarà sobre la informació rebuda d'aquest, per canviar la posició d'aquest, mentre l'acció d'agafar es troba en curs. Per una millor experiència s'utilitzarà la propietat del component *Rigidbody* anomenada *Use Gravity* on es desactivarà la gravetat quan s'agafi l'objecte i s'activarà quan no s'agafi. D'aquesta manera s'obra la possibilitat a tirar un objecte des de certa altura i que quan se solti, es vegi afectat per la gravetat de l'escena.

Per implementar la manipulació d'agafar sobre l'objecte a modelar, s'implementarà un *script* que realitzi les comprovacions i mètodes necessaris per dur a terme aquesta interacció (vegeu apèndix A.3). En aquest *script* s'obtindrà el *GameObject* per mitjà del *Raycast* i si aquest és l'objecte a modelar, s'anirà canviant de posició en l'escena tenint en compte les mans de l'avatar, tot això en funció del valor de la variable *isGrabbing*.

Una vegada agregat aquest script en les dues mans de l'avatar, quan es tanquin les mans es crearà un vincle entre l'extremitat i l'objecte a modelar, aquest no se soltarà fins que no s'obri la mà. L'objecte seguirà en tot moment la posició de la mà emulant l'acció com si s'hagués agafat realment (vegeu figura 4.24). Tant la rotació, com la mida de la figura no es veuran afectats per aquest tipus de manipulació, ja que únicament aquesta acció canvia la posició de l'objecte en l'escenari tridimensional.

[11] Physics.Raycast. (2019, 15 d'abril). *Unity | Documentation*. Data de consulta: 13:42, maig 3, 2019 des de https://docs.unity3d.com/ScriptReference/Physics.Raycast.html.


FIGURA 4.24: Agafament de l'objecte mitjançant les extremitats
Font: Creació pròpia

**Deformar la figura I: Introducció**

La deformació d'un objecte en l'escena, en aquest cas una figura, és de les tasques més bàsiques i a la vegada més complexes que existeixen a Unity. El fet de poder deformar una malla de col·lisió juntament amb la textura que forma l'objecte no és una tasca fàcil. El fet de deformar, implica en Unity la creació de nous polígons si s'escau, o l'eliminació d'algun d'ells quan es produeix una col·lisió. Serà necessari calcular entre altres aspectes, la velocitat en què es produeix la col·lisió i l'impacte que pot causar a la figura, ja que a major velocitat més impacte i viceversa.

A l'hora de deformar la figura serà imprescindible que les mans de l'avatar siguin capaces de crear col·lisions contra la malla de col·lisió de la figura. Aquest fet hauria de provocar una lleugera deformació, depenent de la velocitat de l'impacte, en la figura que s'està modelant.

Quan es parla de col·lisions, si es recorda, han estat utilitzades en la Secció 4.2.3 al final de l'anterior iteració. Que s'intenta explicar amb això? Doncs bé, si s'afegeix la deformitat d'una figura, la interacció per deformar-la serà mitjançant la col·lisió amb la figura. Anteriorment aquesta interacció ja venia implementada per poder moure la figura sense la necessitat d'agafar-la, per tant l'agregació d'aquesta nova manipulació en la figura provocarà que se solapin el nombre d'accions sobre la figura en una mateixa interacció. En aquest projecte no es prescindirà de cap acció que hagi estat implementada a la figura anteriorment, per tant se solaparan les interaccions intentant deformar i moure la figura amb el realisme més gran possible.

Per afegir aquesta nova acció sobre la figura, s'ha dut a terme una cerca exhaustiva sobre les diferents alternatives que hi poden haver, ja sigui mitjançant la importació d'*assets* externs, com la realització de possibles *scripts* per a completar l'objectiu.

En la utilització d'*assets* externs, seran *scripts* proporcionats per diversos autors els que proporcionaran aquesta funcionalitat[29][30], únicament serà necessari adaptar-ho a les exigències que requereix el projecte per fer-lo funcionar. A continuació, es veuran les diferents formes de deformar.

### Meshinator

Es tracta d'un paquet de *assets* extern i gratuït que permet la deformació per mitjà de col·lisions en l'objecte el qual s'adjuntarà el *script*. Així mateix, també proporciona diverses característiques com la resistència del material, la força en la qual es produeix l'impacte, el tipus d'impacte que causarà (deformació de l'objecte o fractura), etc. Alguns dels problemes principals és la utilització de mètodes obsolets, ja que la creació d'aquest *asset* es va dur a terme en versions molt antigues d'Unity.

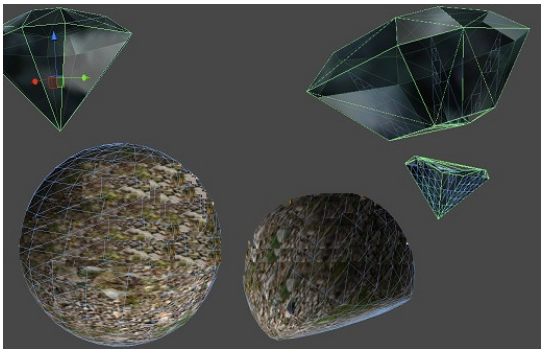


FIGURA 4.25: Possibles resultats en la utilització de l'*script* Meshinator
Font: https://assetstore.unity.com

El paquet d'*assets* Meshinator funciona amb l'impacte que causarà l'objecte en col·lisionar una vegada hagi entrat dins de l'àrea de col·lisió de l'objecte el qual s'interaccionarà (similar a modelar una placa fina de ferro). Això serà un factor determinant a l'hora d'elegir la forma d'implementar-ho posteriorment, ja que s'adaptarà millor o pitjor segons l'exigència que requereix el mateix projecte. Un exemple d'aquest *asset* seria el de l'anterior figura 4.25.

### Impact Deformable

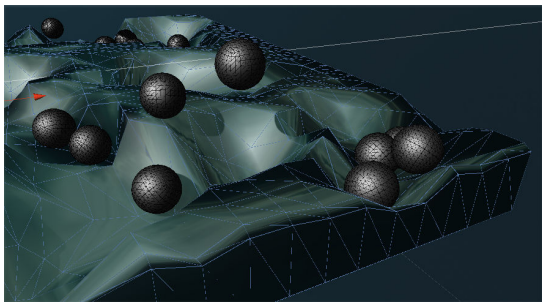


FIGURA 4.26: Possibles resultats en la utilització de l'*script* Impact Deformable
Font: http://www.arcadiumplayware.com

És un paquet d'*assets* similar a l'explicat anteriorment, on en aquest cas sí que no és gratuït. Disposa de diversos escenaris d'exemple per poder comprovar la seva utilitat i a més, disposa de documentació per a poder-lo utilitzar correctament. El mètode d'agregació del *script* és el mateix que en el *Meshinator*, s'afegirà el *script* en l'objecte el qual es produirà aquesta deformació. El *script* està actualitzat en les últimes versions d'Unity i no disposa de problemes en la utilització de mètodes obsolets.

En aquest cas, Impact Deformable funciona tant amb la col·lisió d'entrada, com en la col·lisió una vegada s'està dins de l'àrea. En aquest tipus de col·lisió es crearà deformació tant si es produeix l'impacte inicialment, com si es manté l'objecte col·lisionant en el temps, sense sortir de l'àrea de col·lisió (vegeu figura 4.26).

**Deformar la figura II: Procediment**

Les dues solucions proposades en les anteriors seccions són perfectament vàlides per dur a terme el modelatge de la figura, però aquí apareix el tercer problema i més complicat de tots i fa referència a les col·lisions que vénen causades per les mans de l'avatar.

Com s'ha configurat el Rigidbody de les dues mans de l'avatar, totes dues no estan afectades per la força de la gravetat, però sí que estan afectades per scripts externs, com els que s'han introduït a la secció 4.1.3, per poder donar animació a l'avatar i per a imitar els moviments de l'usuari. Doncs bé, aquesta és la propietat principal del *isKinematic*, l'ús d'aquesta propietat implica que l'objecte en si no exerceix cap mena de força i per tant quan col·lisionarà amb la figura no tindrà cap impacte. Els *scripts* explicats anteriorment, treballen tots dos sobre col·lisions, és a dir, per un funcionament correcte seria necessari que l'objecte que col·lisiones exerceixi alguna mena de força com podria ser, per exemple, la força de la gravetat.

Per solucionar-ho, caldrà exercir alguna força a partir del moviment o d'algun paràmetre que sigui capaç de provocar alguna mena de força per a generar una col·lisió artificial, sigui fictícia o almenys que sigui coherent amb la força de l'impacte. La solució, en aquest cas, implicarà controlar quan s'entra i es manté la col·lisió, per tal de poder anar actualitzant la força de l'impacte en cada moment.

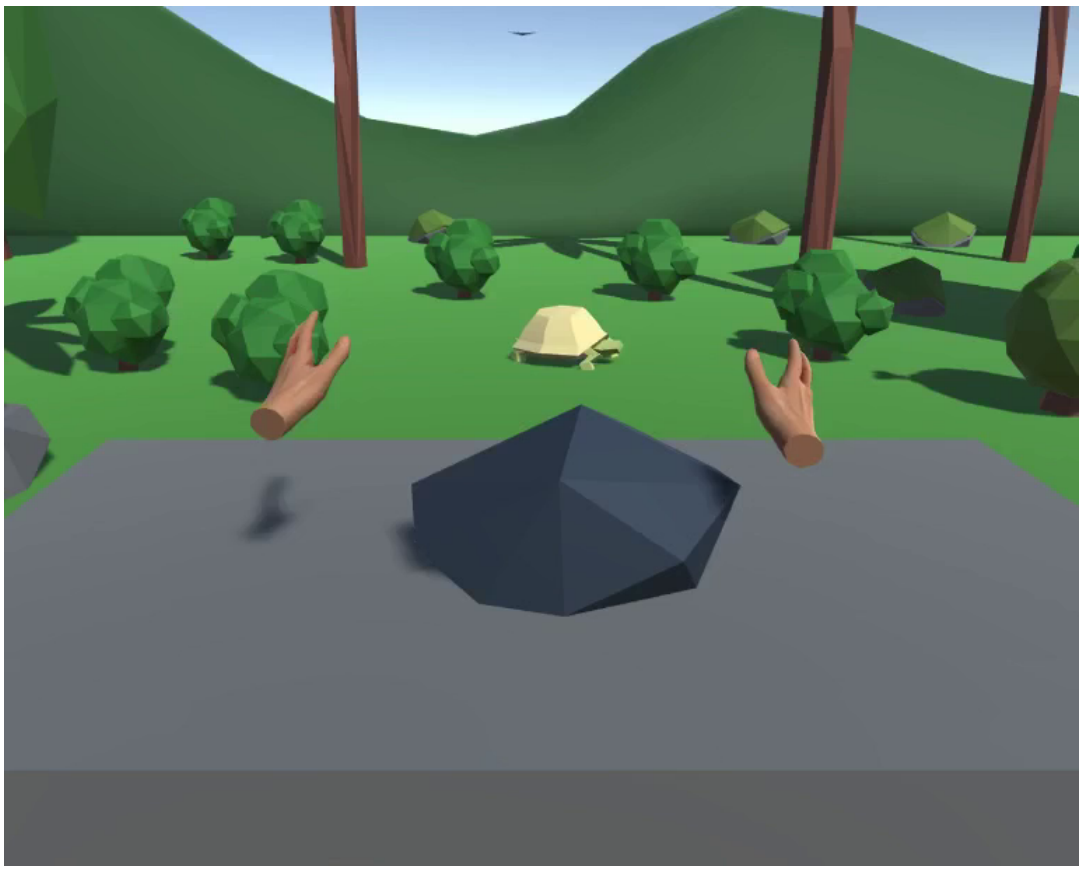


FIGURA 4.27: Resultat obtingut amb la deformació de la figura (GIF)
Font: Creació pròpia

El fet de poder guardar dues posicions en diferents intervals de temps, serà suficient per a calcular la diferència de les coordenades en l'espai tridimensional de les dues posicions en l'instant $x$ de temps. Així mateix, aquesta diferència de coordenades, serà suficient per assignar-la a la col·lisió fictícia quan es produeixi, ja que com major sigui l'interval, més velocitat de moviment en la col·lisió hi haurà i per altra banda, contra menor sigui l'interval, significarà que el desplaçament ha estat mínim i per tant la força de col·lisió serà baixa o pràcticament nul·la.

El resultat que s'ha obtingut mitjançant el mètode de crear col·lisions amb forces fictícies es pot observar en l'anterior figura 4.27. On s'observa que quan les mans de l'avatar fan contacte amb la figura a modelar, aquesta es deforma en funció del seu impacte.

#### Mode de pausa

Per acabar amb el desenvolupament de totes les funcionalitats d'aquest prototip, s'implementarà un mode de pausa perquè sigui possible una comunicació amb l'altre mòdul del sistema que fa referència a les interfícies del prototip. Així doncs s'ha creat un nou *script* per fer-ho possible on la finalitat d'aquest, és executar-se un esdeveniment quan l'usuari tanqui les dues mans al mateix temps. La idea és que surti un menú de pausa, que estarà dissenyat prèviament per l'altre mòdul.

Per la implementació d'aquesta funcionalitat s'ha implementat el següent *script* que apareix en l'apèndix A.4 on de moment, llançarà un missatge de depuració un cop es produeixi aquest esdeveniment. S'implementarà així, ja que aquesta tasca no formarà part d'aquest mòdul, únicament es deixarà la funcionalitat implementada per a una futura integració.

### 4.3.4 | Dificultats trobades

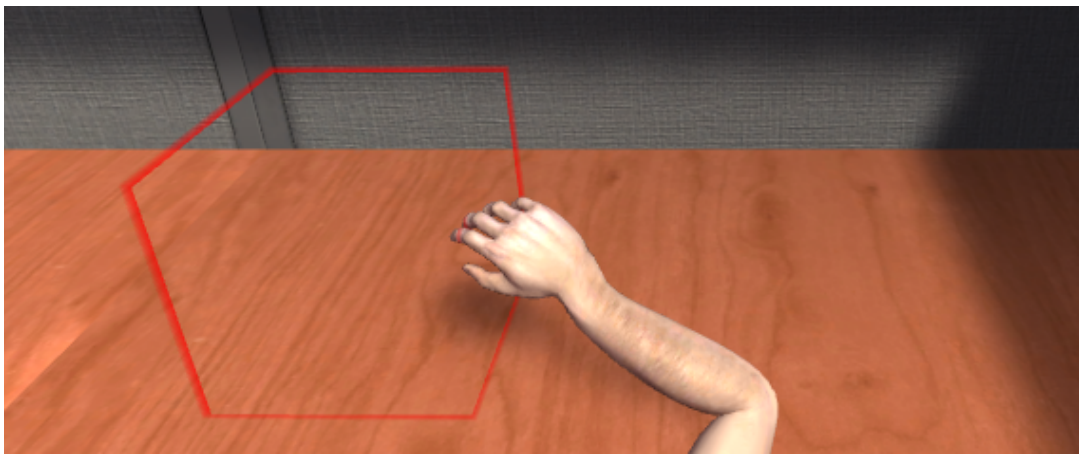


FIGURA 4.28: Desaparició de la textura quan s'utilitza el remarcat de la figura en produir-se un canvi de posició
Font: Creació pròpia

Les dificultats que s'han trobat en aquesta tercera i última iteració fan referència sobretot en els problemes que apareixen respecte a la incorporació del ressaltat en la figura a modelar. En la secció 4.3.3 quan es va implementar la funció d'agafar la figura, aquesta complia els resultats esperats del ressaltat però, el problema venia quan es produïa el desplaçament de la figura sobre l'escena tridimensional. Aquesta perdia la seva textura i deixava de ser visible per l'usuari que interaccionava. La causa venia provocada bàsicament per l'*script* que generava el ressaltat de la figura, encara que es desconegui quina era la causa amb precisió d'aquest problema, únicament es sap que quan es prescindia dels *scripts* referents al ressaltat, es podia interaccionar amb la figura amb total normalitat i deixava de sorgir aquest problema (vegeu figura 4.28).

FIGURA 4.29: Males deformacions que apareixen en la figura a modelar i que són incoherents amb les col·lisions
Font: Creació pròpia

Per altra banda, existeixen alguns problemes de disseny que van relacionats amb la deformació de la figura, més concretament quan aquesta ha variat en gran mesura la seva forma. Ja que existeixen diversos punts, en funció de la seva deformació que són a vegades incoherents segons l'impacte que ha estat produït prèviament (vegeu figura 4.29).

Les dificultats més importants a l'hora de desenvolupar aquesta tercera iteració fan referència en l'ús de mètodes per agafar la figura (vegeu secció 4.3.3). Principalment, els mètodes d'implementació que s'han dut a terme, ha estat mitjançant l'ús de *Raycast*. Encara que ha estat gairebé impossible cercar referències que expliqués l'ús del *raycast* que no fos amb el ratolí per haver de desplaçar un objecte.

Per acabar, una dificultat que no s'ha pogut resoldre fins al moment, és la modificació dels *colliders* quan l'objecte al que està associat canvia de forma, és a dir, en les mans de l'avatar, quan es produeix l'animació d'obrir i tancar la mà, el *collider* associat en cada mà no varia de forma, produint així que si es mou o deforma la figura amb la mà tancada, es podria produir una col·lisió que realment no haurà de passar (vegeu figura 4.30).

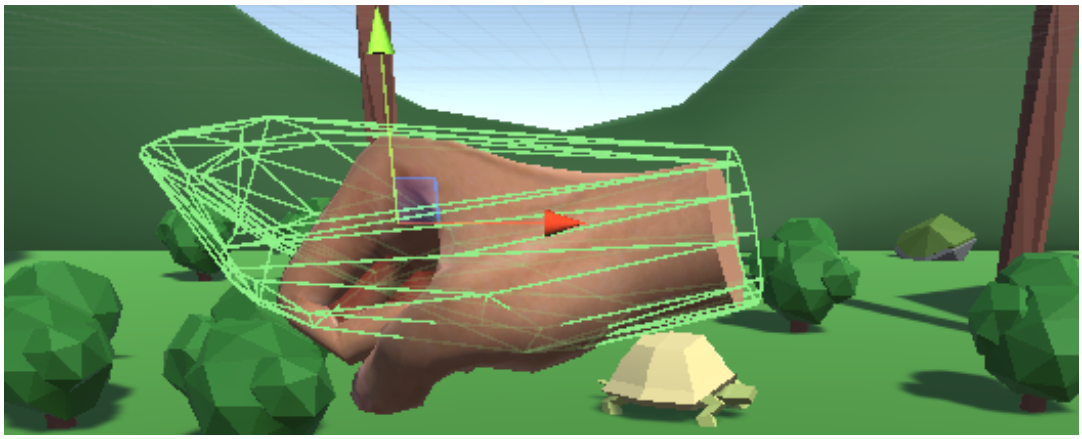


FIGURA 4.30: El *collider* no rep una deformació quan la figura canvia de forma, en aquest cas la mà
Font: Creació pròpia

Aprofitant que sorgien els problemes dels canvis de postura amb els altres tipus d'avatars en les anteriors iteracions, s'ha implementat l'ús del *Mesh collider* per tal d'aportar més realisme a l'hora de col·lisionar quan es produeix la interacció amb l'objecte. Tenint en compte que ara únicament s'està treballant a nivell de mans, s'-ha vist una bona alternativa implementar aquest tipus de malla de col·lisió, ja que exceptuant el problema esmentat anteriorment, les interaccions són més realistes.

# 5 Conclusions i línies futures

## 5.1 | Conclusions

Fent referència a l'experiència personal sobre aquest projecte, esmentar tota la implicació que s'ha dut a terme sobre aquest camp de la informàtica com és la interacció persona-ordenador. No ha estat un projecte fàcil per la manca de referències en moltes parts del desenvolupament, pel poc coneixement que es tenia d'aquest tema i de tot el que això comporta. En general, ha estat un procés més d'investigació que de desenvolupament, ja que en comparació a les hores invertides, que no han estat poques, el resultat que s'ha obtingut d'aquest mòdul ha estat l'assoliment de tots els objectius plantejats a l'inici del projecte.

Al llarg d'aquesta memòria s'ha presentat, com s'ha dit des d'un bon principi, la fase inicial d'un projecte centrat en la interacció basada en moviment on l'objectiu final és la manipulació d'objectes en entorns 3D de manera no invasiva per l'usuari. Això pot ajudar en un futur a proporcionar solucions en diferents àmbits, com les activitats esportives o la medicina, permeten que diferents usuaris no hagin de fer ús d'un dispositiu (que transportin) i de moviments quotidians evitant càrrega mental que sí que requereixen altres mecanismes d'interacció.

La manipulació suposa, en aquest cas, modificar la posició, orientació i forma d'un objecte per mitjà de l'ús d'un dispositiu, en aquest cas la càmera Orbbec Astra Pro juntament amb la SDK de Nuitrack. Això ha permès a l'usuari, interactuar a través del moviment de les seves extremitats superiors, més concretament, les seves mans. Aquest projecte s'ha centrat en l'etapa inicial, consistent a la investigació requerida a l'elecció del sensor 3D, i també la creació de l'entorn 3D per la seva interacció. Tanmateix, la tasca principal a destacar, ha estat el treball realitzat respecte a la interacció entre l'usuari i l'objecte tridimensional, fet que implica tres processos essencials: monitoratge, seguiment de les articulacions de l'usuari i reconeixement de gesticulacions.

La implementació en el prototip d'elements amb una aportació destacable com és la compatibilitat amb diferents sensors, fan que aquest prototip suposi una alternativa en el món de la interacció basada en moviment sense elements invasius per l'usuari.

En aquest aspecte, el desenvolupament sobre el qual s'ha treballat i explorat, permet incorporar noves tecnologies sense la necessitat de desfer-se d'una implementació ja realitzada. Aquest fet ajuda a l'hora de no limitar l'ús del sistema en el temps, com actualment passa amb altres tecnologies, tals com l'ús del sensor Kinect, on el seu desenvolupament ha estat parat des de fa temps.

La interacció amb un objecte tridimensional fent ús d'articulacions o extremitats de l'usuari, com les mans, permet elaborar entorns, on el seu ús va més enllà de la simple exploració de la manipulació d'objectes. El resultat ajuda a conèixer la capacitat d'ajudar en camps tan essencials avui en dia, com l'entrenament de capacitats específiques d'usuaris, millora d'habilitats, etc.

S'ha observat en el desenvolupament del sistema i les diverses proves que s'han dut a terme, que les preparacions generals que aporta el desenvolupador respecte a l'ús de la càmera 3D són de vital importància, ja que l'incompliment d'alguna preparació pot causar imprecisions en el sistema i això aporta una mala experiència de l'usuari, a part de la resposta errònia que pot causar.

## 5.2 | Treball futur

Respecte al treball futur, s'ha considerat de vital importància la utilització d'altres articulacions de l'usuari (per exemple, cames o braços) permetent així, afegir més punts d'interacció entre l'usuari i l'objecte a manipular. En un futur es podria treballar en la incorporació de la capacitat de fer ús dels dits de la mà, oferint així, nous punts d'interacció. Això suposa un repte fer ús del reconeixement de gesticulacions tant corporals, com facials, per afegir més possibilitats i ampliar la utilitat del sistema final.

La incorporació d'elements externs o dispositius d'entrada, com per exemple uns guants de realitat virtual, suposaria un increment molt notori amb la interacció, ja que implicaria un increment en la precisió dels moviments amb les mans. Tot això seria a canvi de la pèrdua de la interacció no invasiva, ja que es disposaria d'elements que entrarien en contacte amb l'usuari. Amb aquesta nova incorporació, obra un món de possibilitats a dur a terme, ja sigui per l'entrenament de capacitats específiques, com per un simple entreteniment. La realització d'un prototip on es pogués intercanviar l'objecte a manipular per un cub de Rubik i realitzar-lo, seria tot un repte per aquest tipus d'interacció.

# Bibliografia


[1] J. E. Garrido, V. M. R. Penichet, M. D. Lozano, A. M. Plata, and J. A. F. Valls, "The use of joint coordinates to monitor patients in a movement-based interaction system," *Universal Access in the Information Society*, Nov 2017.

[2] J. A. Fernandez-Valls, A. M. Plata, V. M. R. Penichet, M. D. Lozano, and J. E. Garrido, "Rehabilitación física a partir de interacción basada en movimiento," in *2016 IEEE 11th Colombian Computing Conference (CCC)*, pp. 1–7, Sep. 2016.

[3] "Interfaz de usuario." https://es.ryte.com/wiki/Interfaz_de_Usuario.

[4] A. Robertson, "New lego augmented reality app is the best open - world lego video game." https://www.forbes.com/sites/andyrobertson/2017/12/01/new-lego-augmented-reality-app-is-the-best-open-world-lego-video-game/#46342c03498a, març de 2018.

[5] A. Madruga, "Paradigmas de interacción." https://cibernetico.org/2007/12/07/paradigmas-de-interaccion/, desembre de 2007.

[6] E. Salas, "Nui (natural user interface)." http://qbit.com.mx/blog/2012/03/02/nui-natural-user-interface/, març de 2012.

[7] A. M. Guerrero and C. E. Jaramillo, "Entorno de inmersión visual interactivo," Sep. 2017.

[8] V. Pterneas, "Kinect is dead! now what?." https://pterneas.com/2017/10/25/kinect-dead/, octubre de 2017.

[9] A. D. C. A. Coroiu and A. Coroiu, "Interchangeability of kinect and orbbec sensors for gesture recognition," in *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 309–315, Sep. 2018.

[10] I-Programmer, "Openni to close." https://www.i-programmer.info/news/194-kinect/7004-openni-to-close-.html, febrer de 2014.

[11] 3DiVi, "Overview: What is nuitrack$^{TM}$sdk?." https://download.3divi.com/Nuitrack/doc/Overview_page.html, desembre de 2018.

[12] 3DiVi, "Zombie nightmare (oculus rift)." https://download.3divi.com/Nuitrack/doc/UnityZombies_page.html, desembre de 2018.

[13] 3DiVi, "Nuitrack sdk architecture." https://download.3divi.com/Nuitrack/doc/Architecture_page.html, desembre de 2018.

[14] 3DiVi, "General preparations." https://download.3divi.com/Nuitrack/doc/Preparations_page.html, desembre de 2018.

[15] J. Webb and J. Ashley, *Skeleton Tracking*, pp. 85–120. Berkeley, CA: Apress, 2012.

[16] J. Donzet, E. Fernández, and R. Leira, "Tecnologías de interacción avanzadas aplicadas a videojuegos," *Blucher Design Proceedings*, vol. 1, no. 8, pp. 149 – 152, 2014.

[17] V. Pterneas, "Body tracking using orbbec astra + nuitrack (kinect alternative)." https://pterneas.com/2018/04/30/orbbec-astra-nuitrack/, abril de 2018.

[18] 3DiVi, "Unity tutorials." http://download.3divi.com/Nuitrack/doc/UnityTutorials_page.html, desembre de 2018.

[19] Orbbec, "Develop with orbbec." https://orbbec3d.com/develop/, abril de 2019.

[20] J. Mula, "Pros y contras de programar en unity vs. en unreal engine." https://www.deustoformacion.com/blog/diseno-produccion-audiovisual/pros-contras-programar-unity-vs-unreal-engine, novembre de 2017.

[21] C. Cat, "3d scifi kit starter kit." https://assetstore.unity.com/packages/3d/environments/3d-scifi-kit-starter-kit-92152, juliol de 2018.

[22] Invector, "Third person controller - basic locomotion free." https://assetstore.unity.com/packages/templates/systems/third-person-controller-basic-locomotion-free-82048, desembre de 2017.

[23] 3DiVi, "Nuitrack skeleton tracking." https://assetstore.unity.com/packages/templates/packs/nuitrack-skeleton-tracking-127675, gener de 2019.

[24] Unknown, "First person arms." http://conceptforart.blogspot.com/2013/11/blog-post_28.html, novembre de 2013.

[25] Tsunami, "Office mega kit - free download." http://unityassetcollection.com/office-mega-kit-free-download/, setembre de 2018.

[26] Tsunami, "Highlighting system – free download." http://unityassetcollection.com/highlighting-system-free-download/, setembre de 2018.

[27] Tsunami, "Hands for vr: Basic – free download." http://unityassetcollection.com/hands-for-vr-basic-free-download/, agost de 2018.

[28] K. Cederlund, "Click on object with raycast screen - unity [eng]." https://www.youtube.com/watch?v=oEywwHERy1U, juny de 2017.

[29] MD, "Meshinator - real-time mesh deformation tool." http://meshinator.blogspot.com/, abril de 2013.

[30] L-Tyrosine, "Impact deformable." http://www.arcadiumplayware.com/home/impact-deformable/, juliol de 2018.

# A Scripts

En les següents seccions de l'annex A es podrà trobar els diferents *scripts* creats en el llenguatge de programació C Sharp i utilitzats en Unity 3D per al desenvolupament del projecte.

## A.1 | MoveAvatar.cs

```
using UnityEngine;
using System.Collections.Generic;

public class MoveAvatar : MonoBehaviour
{
    [Header("Rigged model")]
    [SerializeField]
    ModelJoint[] modelJoints;

    /* Diccionari Key: articulacio objectiu, Value: estructura art. */
    Dictionary<nuitrack.JointType, ModelJoint> jointsRigged =
        new Dictionary<nuitrack.JointType, ModelJoint>();

    void Start()
    {
        for (int i = 0; i < modelJoints.Length; i++) {
           modelJoints[i].baseRotOffset = modelJoints[i].bone.rotation;
           jointsRigged.Add(modelJoints[i].jointType, modelJoints[i]);

           /* Si te un os pare, calcular les distancies */
           if (modelJoints[i].parentJointType!=nuitrack.JointType.None)
               AddBoneScale(modelJoints[i].jointType,
                            modelJoints[i].parentJointType);
        }
    }

    void AddBoneScale(nuitrack.JointType targetJoint,
                      nuitrack.JointType parentJoint)
    {
       /* Agafar la posicio de l'os a moure i el seu pare */
       Vector3 targetBonePos = jointsRigged[targetJoint].bone.position;
       Vector3 parentBonePos = jointsRigged[parentJoint].bone.position;
```

```
        jointsRigged[targetJoint].baseDistanceToParent =
                        Vector3.Distance(parentBonePos, targetBonePos);

        /* Guardar transformacio de l'os pare i extrure de jerarquia */
        jointsRigged[targetJoint].parentBone = jointsRigged[parentJoint]
                                                .bone;
        jointsRigged[targetJoint].parentBone.parent = transform.root;
    }

    void Update()
    {
        if (CurrentUserTracker.CurrentSkeleton != null)
            ProcessSkeleton(CurrentUserTracker.CurrentSkeleton);
    }
    void ProcessSkeleton(nuitrack.Skeleton skeleton)
    {
        foreach (var riggedJoint in jointsRigged) {
            /* Obtenir l'articulacio de Nuitrack i de l'avatar*/
            nuitrack.Joint joint = skeleton.GetJoint(riggedJoint.Key);
            ModelJoint modelJoint = riggedJoint.Value;

            /* Posicio de l'os objectiu */
            Vector3 newPos = Quaternion.Euler(0f, 180f, 0f) *
                                        (0.001f * joint.ToVector3());
            modelJoint.bone.position = newPos;

            /* Rotacio de l'os objectiu */
            Quaternion jointOrient = Quaternion.Inverse(
                                    CalibrationInfo.SensorOrientation)
                                    * (joint.ToQuaternionMirrored())
                                    * modelJoint.baseRotOffset;
            modelJoint.bone.rotation = jointOrient;

            /* Escalar l'os del pare si n'hi ha */
            if (modelJoint.parentBone != null)
            {
                Transform parentBone = modelJoint.parentBone;

                /* Calcular la distancia actual respecte la base */
                float scaleDif = modelJoint.baseDistanceToParent /
                        Vector3.Distance(newPos, parentBone.position);

                /* Canviar la mida de l'os */
                parentBone.localScale = Vector3.one / scaleDif;
            }
        }
    }
}
```

## A.2 | Animation.cs

```
using UnityEngine;

public class Animation : MonoBehaviour
{
    public enum Hands { left = 0, right = 1 };

    // Gesture recognition
    [SerializeField]
    Hands currentHand;

    // Save animations
    Animator anim;
    int grab = Animator.StringToHash("Grab");
    int idleStateHash = Animator.StringToHash("Base Layer.Idle");

    // Start is called before the first frame update
    void Start()
    {
        NuitrackManager.onHandsTrackerUpdate +=
            NuitrackManager_onHandsTrackerUpdate;
        anim = GetComponent<Animator>();
    }

    void OnDestroy()
    {
        NuitrackManager.onHandsTrackerUpdate -=
            NuitrackManager_onHandsTrackerUpdate;
    }

    private void NuitrackManager_onHandsTrackerUpdate(nuitrack.
        HandTrackerData handTrackerData)
    {
        bool active = false;
        bool press = false;

        foreach (nuitrack.UserHands userHands in handTrackerData.
            UsersHands)
        {
            if (currentHand == Hands.right && userHands.RightHand !=
                null)
            {
                active = true;
                press = userHands.RightHand.Value.Click;
            }
            else if (currentHand == Hands.left && userHands.LeftHand !=
                 null)
            {
                active = true;
                press = userHands.LeftHand.Value.Click;
            }
        }
        anim.SetBool("isGrabbing", active && press ? true : false);
    }
}
```

## A.3 | DragRigidbody.cs

```
using UnityEngine;

public class DragRigidbody : MonoBehaviour
{
    private Animator animator;
    private bool grabObj = false;
    private GameObject hitObj;

    private void Start()
    {
        animator = GetComponent<Animator>();
    }

    private void Update()
    {
        var mainCamera = FindCamera();
        RaycastHit hit = new RaycastHit();

        Vector2 pointOnScreenPosition = mainCamera.WorldToScreenPoint(
            transform.position);
        pointOnScreenPosition = new Vector2(pointOnScreenPosition.x,
            Screen.currentResolution.height - pointOnScreenPosition.y);

        if (!animator.GetBool("isGrabbing") && grabObj == true)
        {
            grabObj = false;
            hitObj.GetComponent<Rigidbody>().useGravity = true;
        }

        if (Physics.Raycast(transform.position, transform.forward, out
            hit, 5))
        {
            if (!hit.rigidbody || hit.rigidbody.isKinematic)
                                    return;

            if (hit.collider.gameObject && animator.GetBool("isGrabbing
                ") && grabObj == false)
            {
                hitObj = hit.collider.gameObject;
                hitObj.GetComponent<Rigidbody>().useGravity = false;
                grabObj = true;
            }
        }

        if (grabObj)
            hitObj.transform.position = new Vector3(gameObject.
                transform.position.x,
                    gameObject.transform.position.y,
                    (float)(gameObject.transform.position.z + 0.3));
    }

    private Camera FindCamera()
    {
        if (GetComponent<Camera>())
            return GetComponent<Camera>();

        return Camera.main;
    }
}
```

## A.4 | PauseMode.cs

```
using UnityEngine;

public class PauseMode : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {
        NuitrackManager.onHandsTrackerUpdate +=
            NuitrackManager_onHandsTrackerUpdate;
    }

    void OnDestroy()
    {
        NuitrackManager.onHandsTrackerUpdate -=
            NuitrackManager_onHandsTrackerUpdate;
    }

    private void NuitrackManager_onHandsTrackerUpdate(nuitrack.
        HandTrackerData handTrackerData)
    {
        foreach (nuitrack.UserHands userHands in handTrackerData.
            UsersHands)
        {
            if (userHands.RightHand != null && userHands.LeftHand !=
                null)
            {
                if (userHands.RightHand.Value.Click && userHands.
                    LeftHand.Value.Click)
                {
                    // PAUSE MODE
                    Debug.Log("PAUSE MODE!");
                }
            }
        }
    }
}
```